Archived Material

Historical Purposes Only

Workshop on Software Tools for HPC Systems

Final Report

In October 1996, we organized the sixth in a series of workshops on software tools for high-performance computing. With funding from the DoE, NASA, NSF, and DARPA, the goal of this workshop was to explore the feasibility of creating a national software tool testing and evaluation center.

Table of Contents:

Organizers

Ann Hayes
Computing & Communications Division
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Jeffrey Brown
Computing & Communications Division
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Margaret L. Simmons
National Coordination Office
4201 Wilson Boulevard
Arlington, Virginia 22230

Daniel A. Reed
Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Executive Summary

In October 1996, we organized the sixth in a series of workshops on software tools for high-performance computing. With funding from the Department of Energy, the National Aeronautics and Space Administration, the National Science Foundation, and the Defense Advanced Research Projects Agency, the goal of this workshop was to explore the feasibility of creating a national software tool testing and evaluation center. The workshop and its theme were motivated by a growing realization that the small size of the high-performance computing market and the rapid pace of technological change have rendered traditional mechanisms for software technology transfer ineffective.

Despite continued requests from users, both hardware vendors and independent software developers see little financial incentive for software tool development; tool development costs simply cannot be amortized across sales of tens of high-end systems. Similarly, the research community lacks the financial resources to harden and support valued research prototypes for a large external user base. A software testing and evaluation center offers a possible solution to this conundrum. By identifying and supporting promising software prototypes, a center could serve as a conduit for technology transfer to vendors and as a support center for valued, though not commercially viable, software tools.

To explore the issues surrounding possible creation of a software testing and evaluation center, roughly 100 participants, drawn from academia, government, and industry, met for three days in October 1996 at Cape Cod. The workshop was organized as a sequence of sessions that included invited presentations and extensive, directed discussion periods. In turn, the invited presentations were structured as a mix of technical summaries of current software tool research issues and reviews of software tool distribution mechanisms. The discussion periods focused on both new research problems in debugging and performance analysis and on the organizational structure of a possible software testing and evaluation center.

The workshop participants all agreed that collaboratively developing robust, user-friendly performance tools for high-performance parallel systems is expensive and intellectually challenging. Moreover, effective tool development requires many mundane activities not normally associated with academic computer science research, namely testing early prototypes with friendly users who are application scientists, developing and testing intuitive user interfaces, and writing manuals and documentation.

The workshop attendees concluded that a creating a national, though distributed software tool evaluation and testing center was the only practical solution to the HPCC software tool crisis. Co-located with extant HPCC centers and organized as a cooperating group of developers and support staff, the proposed center would work with academic researchers, application scientists, and vendors to evaluate and test prototype software tools. The center staff would "harden" and

support modest number of prototype tools, creating documentation, porting software to multiple platforms, and fostering standard interfaces for tool interoperability.

By organizing the center as a set of cooperating groups at HPCC sites, the center structure would capitalize on diverse resources and users and, ideally, capture the best features of current distributed centers. Moreover, the center would fill an important gap between the generation of ideas by the research community and consortia like the Parallel Tools (PTOOLS) group and the dissemination of software by the National High-Performance Software Exchange (NSHE).

Background and Motivations

Though fierce vendor competition for a share of the High-Performance Computing and Communications (HPCC) market has led to rapid architectural innovation and higher peak hardware performance, the software infrastructure for massively parallel systems has not kept pace with the rapid evolution of new parallel architectures. The pace of hardware innovation has stretched academic, laboratory, and vendor software development groups to the limit, forcing them to continually create software tools for changing programming models and new hardware environments.

The critical importance of robust, flexible, and efficient software tools has long been recognized and has been a major discussion topic at several national HPCC meetings, including those in Pasadena [3,4] and Pittsburgh [9] and at a series of workshops on software tools hosted by the authors [6,7,8]. These meetings focused on the intellectual difficulties associated with building good tools and, equally importantly, the lack of incentives and infrastructure for tool development and testing.

Research ideas can be explored with a small cadre of graduate students, but building robust tools requires both experience with real user needs and a major development effort. Small profit margins and short product life cycles, both leading to small installed product bases, have made it extraordinarily difficult for tool developers to gain the needed experience with user software development idioms and to create and support effective tools.

By necessity, tools embody knowledge of the execution environment, identifying performance bottlenecks or logical program errors in terms of application code constructs and their interaction with the execution environment. Because the root causes of poor performance or unexpected program behavior may lie with run-time libraries, compilers, the operating system, or the hardware, tools must gather and correlate information from many sources. Not only does this correlation require interfaces for information access, the need for such information is most often gained only from experience. Experience comes with time, as tool developers understand the common programming idioms, the interactions of application code and the underlying hardware and software, and the user interfaces best suited for relating these interactions in intuitive ways.

Simply put, developing good tools takes time, experience and substantial effort; small profit margins and short product life cycles, both leading to small installed product bases, have made it extraordinarily difficult for tool developers to create and support effective tools.

Poor Software Tools: An Old Problem

Is the dearth of effective software tools a new problem? No, the lack of good software development tools can be traced to the very origins of high-performance computing (e.g., early CDC FORTRAN compilers were very inefficient, and the Cray 1 was delivered with essentially no software). However, as high-performance systems have become more widely available and accessible, the long-standing lack of tools has become more evident. More perniciously, the ferment in the HPCC market has led to short product life cycles, with vendors and application developers repeatedly moving to new architectures and programming models. Often, by the time system software has sufficiently stabilized and there is sufficient knowledge of common programming idioms to permit construction of robust and useful tools, the underlying hardware is no longer performance competitive.

Finally, the burgeoning personal computer market continues to raise user expectations to ever higher levels. The widespread availability of high quality, inexpensive desktop software leads users to (a) question the lack of similar tools on high-performance systems and (b) expect interoperability between desktop tools and those on high-performance systems.

In short, the software infrastructure for massively parallel systems has been unable to keep pace with the rapid evolution of new parallel architectures. The paucity of basic software tools for application program development, debugging, and performance tuning has limited the use of scalable parallel systems to a small cadre of hardy pioneers willing to brave a wilderness of experimental hardware, immature software, and frequently changing tools.

Substantial research money is directed toward study and development of software tools, but as Pancake asked at our 1993 workshop on performance tools, Why don't users use the tools that tool developers develop?" The workshop attendees agreed that the fundamental reasons for this dilemma relate to the lack of incentives for tool development and testing.

First, the massively parallel systems market is small, and the real cost of commercial tool development is high. As vendors prioritize development of new systems, functioning hardware, operating systems and compilers receive the highest priority, for machines cannot be shipped without these. However, competitive pressures and development schedule slippage often lead vendors to ship their new systems prematurely, before other infrastructure such as software development tools can be created. Due to these pressures on vendors to ship, the temptation is very high for them to simply "repackage" internal development tools for external use, even when inappropriate for users.

From a financial perspective, developing robust software tools for a parallel system with projected sales of 500 units is nearly as costly as developing software for the burgeoning personal computer market, but without concomitant financial incentives. These same economics preclude creation of a viable third party software industry (i.e., the independent software vendors (ISVs)). One of the workshop working groups addressed the implications of market size for tool development.

Second, another potential source of software tools, the academic and laboratory research community, lacks the reward structure, the financial resources, and (often) the skills to develop and support robust software tools. The academic computer science community is rewarded, both tangibly and intangibly, for development of software prototypes and publication of the ideas underlying these, but not for the additional development needed to make these prototypes really usable by the larger HPCC community. Moreover, because there is limited interaction between academic tool developers and potential tool users, prototype tools often lack important features or are simply not useful to application developers.

Although there are a plethora of reasons for this situation, most are subsumed under the rubric of limited collaboration and funding. Inappropriate tools often result because application developers, tool developers, and vendors are not intimate collaborators in the tool development process; current academic and commercial realities do not reward such collaboration. Moreover, vendors are reluctant to embrace, extend, and market academic software unless it is already robust, as evidenced by wide use within the application community. Finally, an undue emphasis on peak hardware performance, at the frequent expense of sustained, readily achievable performance, leads to rapid changes in hardware and system software.

Poor Software Tools: A Worsening Problem

In many senses, the software tool dilemma is worsening. The World Wide Web, distributed metacomputing, and new parallel programming models all pose unsolved problems for software tools. Tools for task and data parallel languages, techniques for real-time adaptive system control, and optimization of heterogeneous metacomputing applications. are all united by the need for greater access to system internals and data and by the need for standard interfaces, both internally and for users.

Historically, performance and debugging tools have been developed independently of system software and compilers. This separation reflects both the integration of software from disparate sources (i.e., third party compilers and operating system kernels) and limited support provided by software systems for tool development. For example, operating systems typically provide debugger developers little more than a UNIX ptrace system call for controlling process execution and performance tool developers only a mechanism for accessing a coarse-resolution system clock. Similarly, compilers provide simple symbol table information in object files. For single processor systems, these features are sufficient to build breakpoint debuggers and profilers, though they are far from optimal. For parallel systems and workstation clusters, they are woefully inadequate.

At present, few compilers provide access to program transformation data, though debugging and tuning the performance of programs written in task and data parallel languages (e.g., like High-Performance FORTRAN (HPF) [2]) requires both compile-time and run-time data. Relating the dynamic behavior of compiler-synthesized code to the user's source code requires knowledge of the program transformations applied by the compiler and the code generation model. For example, if an HPF compiler generates message passing code for a distributed memory system, understanding how messages relate to array locality is a key to improving performance. Moving beyond tools for explicit parallelism (e.g., via message passing) will require far tighter integration of tools with compilers.

Likewise, few operating systems or run-time libraries provide mechanisms for selecting resource management policies or for configuring those policies based on knowledge of application resource demands or dynamic performance data. However, many experiments have shown that tuning policies to application behavior is key to improving performance for irregular applications with complex behavioral dynamics. For example, as part of the Scalable I/O Initiative, researchers have shown that selecting and configuring file system input/output policies to match application input/output patterns and hardware configurations can dramatically increase achievable application input/output performance.

The explosion of World Wide Web (WWW) use, together with rapidly expanding interests in distributed data mining and heterogeneous parallel computing, pose a different, though equally thorny, set of tool research problems. New types of debugging and performance tuning tools will be needed to create metacomputing applications that can exploit distributed computation resources (e.g., by distributing a computation across multiple, geographically dispersed parallel systems) and that can mine large data archives in response to complex queries. Measuring network latencies and bandwidths and adapting to changes in network loads will necessitate integration of "standard" tools for parallel systems with distributed network management mechanisms (e.g., like the Simple Network Management Protocol (SNMP)).

Support for new programming models, tuning of resource management policies, and management of distributed metacomputations requires interfaces and access to data not readily available via present methods. This will require creation of a new generation of performance analysis and debugging tools that are more tightly coupled with runtime libraries, operating systems, and compilers. In turn, this greatly exacerbates current software tool development problems (i.e., access to data and software interfaces).

Workshop Context

Recognition of the technical, economic, and sociological problems facing the high-performance computing community motivated us to organize a series of workshops on software tools. The most recent workshop was the sixth in a series of workshops organized by Los Alamos National Laboratory and the University of Illinois. Begun in 1988, these workshops have evolved from an initial focus on the technology and research issues underlying creation of performance and debugging tools to an assessment of the technical, political, and economic constraints on high-performance computing software tools. Simply put, the goal of the workshop series is to address the changing roles and expectations for software tools in the high-performance computing and communications domain.

The first workshop, held in 1988, occurred just as the national High-Performance Computing and Communication (HPCC) program began. At that time, massively parallel systems were the emerging future of high-performance computing, rather than a current reality. A substantial fraction of the first workshop was devoted to directed discussions of performance instrumentation problems on three broad classes of parallel systems architectures: shared memory, distributed memory, and data flow. During these discussions, vendors, academics, and users spoke frankly and at length about the problems they faced and possible solutions [6].

A second workshop was held in 1989 [8]. Here, researchers described the functionality of prototype performance tools and discussed initial experiences applying the tools in real execution contexts. The third workshop, held in 1991, focused on issues of performance measurement, data analysis, and visualization as they related to large-scale parallel systems. By this juncture, real systems had appeared and were being actively used by a growing cadre of software developers.

Reflecting practical experience with software tools and market pressures faced by HPCC vendors, the fourth workshop, held in 1993, shifted its focus to the interplay between technical issues and software "success." The workshop brought together application software developers and tool developers to engage in an exchange of ideas on what are necessary, possible and appropriate performance tools for massively parallel systems. The predominant theme that emerged from the workshop was that software tools, both performance and debugging, were neither widely available nor widely used and that this fact was adversely affecting the entire high-performance computing community.

Finally, the fifth workshop, held in 1994, combined both debugging and performance tool groups [7]. Building on the concerns of the previous workshop, the participants were united in their belief that the only practical solution to the paucity of effective software tools for high-performance systems was a community effort to create a software tool testing center. Such a center would work with academic researchers, application scientists, and high-performance computer vendors to evaluate, test, and enhance prototype software tools. In addition, the center would foster both industry standards for tool interoperability and technology transfer of mature software prototype.

The sixth workshop, summarized in this report, was convened to explore the technical and logistical organization of a possible software testing and evaluation center. The speakers and technical working group topics were chosen to explore these issues in detail. The workshop began with a keynote presentation by Ken Kennedy on technology transfer paths for HPCC software. The talk discussed the "standar"d model of technology push from research groups to both computing vendors and independent software vendors (ISVs) and the economic conditions that have limited its success. Dan Reed then discussed his experiences with software technology transfer, which echoed Kennedy's comments. He then summarized the recommendations of the fifth workshop and its rationale for a possible software testing and evaluation center. Following this, he outlined the organizers' charges to six workshop discussion groups.

Attendees and Student Participation

To explore the issues surrounding possible creation of a software testing and evaluation center, roughly 100 participants, drawn from academia, government, and industry, were invited to attend the workshop. One of the most successful components of the workshop was the inclusion of graduate students. With support from NSF, we invited software tool researchers and one of their graduate students to attend the workshop as a pair. Each student participated fully in the workshop and its discussion groups and also presented a poster on their work at the evening reception.

By involving students in the discussion, we hoped that new research relationships would be established, not only among the researchers, but also among the students. These students are the next generation of computational and computer scientists. It is critical that they form close working relationships; the days when computer science and computational science developed in isolation are past.

Workshop Discussion Groups

Because we have long felt the primary purpose of a workshop should be discussion, not passive listening, roughly half the workshop was devoted to organized discussion via a set of six working groups. Each working group was charged to consider either specific technical issues related to HPCC software tools or a particular set of issues related to organization of a software testing center. These groups addressed the following issues:

- **Technical challenges for performance tools**
  What are the most successful current performance tools today and what are their characteristics? What performance tools will be needed for emerging programming models and new hardware environments such as teraflop and petaflop computing and distributed network computing?

- **Management issues for a software center**
  What are the mission, goals and objectives of a center? Would the center be centralized, distributed, or virtual and how would it be managed in each of these cases?

- **Software selection and vendor involvement**
  What selection criteria should be used to identify the software to be developed at the

center? How might vendors participate in the center's activities and benefit from the center's accomplishments?

- **Intellectual property issues**
  What are the legal/ethical responsibilities, intellectual ownership, and liability issues that must be addressed to establish a software testing center?

- **Technical Infrastructure**
  What technical components (i.e., software, hardware, and staff) would be required by a center? How would the center function and transfer technology?

- **Technical challenges for debuggers**
  What should be the role of a tools center to encourage the development and accelerate the deployment of high quality parallel debuggers for high performance computing?

Working Group Summaries

The appendices to this report contain the analyses of these and other questions as reported by the working groups. Below, we summarize the key points of each working group and the general themes that emerged from the workshop discussions.

Technical Issues (Performance Tools and Debuggers): Working groups one and six considered the technical problems faced by performance tool and debugging researchers, particularly as software tools evolve from homogeneous parallel systems to include distributed collections of heterogeneous systems. The performance tool group reiterated the observation that there is no single model of successful tool development and deployment. Instead, tools evolve in a variety of ways and originate from many sources. In consequence, they suggested that a software testing center should not supplant current mechanisms for tool development and funding, but rather should augment it by evaluating and enhancing research prototypes. They further noted that the primary challenges faced by performance tool developers are heterogeneity, hierarchy, and scale. The emergence of metacomputing, of scalable systems with thousands of processors (e.g., as in the Department of Energy Accelerated Strategic Computing Initiative (ASCI) systems), and of hierarchical shared memory systems and data parallel programming models all pose new problems for performance tool designers.

The debugging working group noted that debugger designers face daunting and conflicting goals: the need to report environment-specific data while also providing a portable, extensible interface. The lack of accepted standards for user interfaces exacerbates tool development and frustrates users who regularly use multiple systems. The group felt that a software testing and evaluation center should foster understanding of user and developer needs via workshops and usability testing. Moreover, the center should encourage standardization, test and evaluate prototypes with real users, and reduce the effort required for vendors and academic researchers to create usable tools. Finally, a center should support useful tools on standard platforms.

center Management Issues: Working group two explored possible management and organizational structures for a testing and evaluation center. The group took as its working hypothesis the belief that the centerÍs goals should be to collect, evaluate, enhance, and support key software tools that have minimal potential for commercialization due to market size. The group felt that this hypothesis implied that the center would necessarily focus on a modest number of prototype tools that had established users and that were sufficiently robust to make extension and testing feasible.

The working group also considered possible management structures for the center. After discussion, the group concluded that the center should incorporate the best features of the distributed NSF center for Research on Parallel Computation (CRPC) and the Corporation for National Research Initiatives (CNRI). The CRPC model for the center would create a distributed set of collaborating partners to evaluate and enhance selected prototype tools. In contrast, a CNRI model would emphasize a small core staff responsible for establishing mini-consortia to evaluate, enhance, and maintain prototypes. Both models have similar goals, differing primarily in the mechanism for forming partners. The group felt that the center should have a permanent staff co-located with a supporting computing infrastructure.

Software Selection and Vendor Participation: Working group three's charter was to explore the criteria used to select software for center development and support and how vendors might participate in the center. After considerable discussion, the group concluded that while a center should encourage vendor participation as much as possible, the primary customers should be the application developers for high-performance systems. This conclusion was based on comments by participating hardware vendors, who noted that hardware generated revenue, not software tools. Hence, any adoption and commercialization of software tools by hardware vendors must be simple and inexpensive. Independent software developers (ISVs) are somewhat more amenable to technology transfer and collaborative software testing and evaluation.

Like other groups, this group assumed that the center would test, evaluate, harden, document, and support software tools. In contrast to other groups, however, they assumed the center might also develop new tools. A formal evaluation board would consider proposals and recommend allocation of center resources to (a) testing and evaluation, (b) prototype tool hardening, and (c) new tool development. This evaluation board would be composed of researchers, vendors, users, and center staff.

The group noted that the mission of the proposed center naturally complements both the Parallel Tools (PTOOLS) consortium and the National HPCC Software Exchange (NHSE). The PTOOLS group works to define, develop and promote parallel tools. As such, it can be viewed as a possible source of prototypes for augmentation by the center. Conversely, the NHSE is a distribution service for software, documents, and data. Hence, it is a logical dissemination mechanism for software produced by the center.

Intellectual Property Issues: The fourth working group considered the implications of software development and augmentation by temporally and spatially distributed groups and the ownership issues implied by such development. Moreover, participants may include academic researchers, vendors, and national laboratory staff, each with intellectual property (IP) rules defined by their employers and funding agencies. The group considered five different models for software and IP management, ranging from an independent testing center to a software hardening model. After much discussion, the group concluded that the software hardening model, similar to that discussed by other groups, was the most viable and useful.

Under the model, the center would accept software testing proposals from the community, identify appropriate prototype tools, enhance and augment the tools, create user documentation, and convert them to usable tools. As with the software selection group, the IP working group identified end users as the primary target rather than HPCC vendors. In consequence, the group assumed that the center would continue to distribute software hardened tools throughout their useful lifetime.

To address the intellectual property problems that arise from multiparty development and technology transfer, the HPCC agencies must adopt consistent IP policies. It also recommended that the center serve as a clearing house for IP information and education for the HPCC community. To minimize later problems, the center should insist on a full IP disclosure on tools prior to their adoption by the center and all IP issues should be negotiated in advance.

Technical Infrastructure: Working group five considered the resources needed by a national software testing and evaluation center. The group took as its working hypothesis the belief that the center would serve five functions: early evaluation of promising ideas, enhancement of research prototypes to usable levels, creation of reference implementations for key tools, definition of standard tool interoperability conventions, and education of researchers and students.

With this basis, the group concluded that a testing and evaluation center must be co-located with one or more national HPCC sites (e.g., Department of Energy laboratories, NSF supercomputer centers, or NASA centers with strong computational programs). To ensure diversity of users and machines, the center should be physically distributed across a small number of HPCC sites. This distribution and co-location would not only enable access to high-performance computing facilities, it would allow center staff to exploit local expertise, both for software testing with real users and interaction with technical support staff. Although co-located, the testing center should be independent of the host organization, with its own funding base and management authority. Finally, the center staff would consist of a group of software tool developers, user support staff, and technical writers.

To support software testing and evaluation, the center would also need dedicated computing facilities, though not of the large scale available at national HPCC centers. As a complement, the center would also require access to commercial software development tools and to source code for key HPCC system components. These "crashable" facilities would enable the testing center staff to evaluate and develop prototypes that interact closely with system software and hardware while not interrupting production computation on large systems.

Workshop Recommendations

The workshop participants all agreed that collaboratively developing robust, user-friendly performance tools for high-performance parallel systems is expensive and as intellectually challenging as national challenge science, yet it is often assumed to be "easy," despite the fact that tool developers on new systems face the same software problems as application developers, and their tools must interoperate with multiple applications. Moreover, effective tool development requires many mundane activities not normally associated with academic computer science research, namely testing early prototypes with friendly users who are application scientists, developing and testing intuitive user interfaces, and writing manuals and documentation. Most of the activities are not rewarded by the computer science research community. Activities such as supporting a user community, teaching training classes, and adapting software to new hardware and software releases will not bring tenure, peer adulation, better graduate students or (usually) larger research support; therefore, most academic computer scientists have little incentive to develop a tool beyond the prototype stage.

The workshop attendees concluded that a creating a national, though distributed software tool evaluation and testing center was the only practical solution to the HPCC software tool crisis. Co-located with extant HPCC centers and organized as a cooperating group of developers and support staff, the proposed center would work with academic researchers, application scientists, and vendors to evaluate and test prototype software tools. The center staff would "harden" and support modest number of prototype tools, creating documentation, porting software to multiple platforms, and fostering standard interfaces for tool interoperability.

These views are not simply those of the workshop organizers or the participants in the most recent workshop. They mirror those expressed at the Pasadena [3,4] and Pittsburgh [9] meetings and at the five performance and debugging tool workshops. The workshop attendees have repeatedly recommended a major project to understand the limitations of current tools, the requirements for future tools, and to exploit this knowledge to transfer useful software prototypes to the application community and to commercial vendors.

Clearly, however, one center cannot and should not supplant traditional mechanisms for technology transfer from academia to industry. We envision a more synergistic relationship with these traditional mechanisms. Moreover, it is important to realize that there are many possible definitions of software tool "success" that do not include commercialization. Indeed, as argued earlier, commercialization is extraordinarily difficult, and many valuable tools have no commercial market. Pragmatically, success means that a software tool is useful, tested, documented, and widely used. Hence, the goal of the center would be to serve as the focal point for the software tool development, vendor, and computational science communities to increase the number and breadth of useful and necessary tools and to encourage commercialization. To realize this goal, the workshop attendees and organizers envision four foci.

Early Prototype Testing and Evaluation: Typically, academic tool researchers develop simple software prototypes to demonstrate a proof of concept prior to publication. Because most of these projects are small, there are few opportunities to test the ideas with appropriate external users and to learn what aspects of the approach have practical merit. By coupling academic tool researchers with a group of friendly users at supercomputing centers, where there is the infrastructure needed to support early prototypes, the tool researchers can gain early feedback on the practicality of their ideas. Those prototypes that show promise of potential usefulness to application developers could then be further developed with the center's assistance, and where appropriate, in cooperation with one or more vendors. An additional bonus from this stage would be also to gain new research ideas.

Mature Prototype Testing and Extension: Smaller numbers of prototypes are sufficiently mature to be used and useful to a user community not co-located with the tool developer. Major reasons for the importance of co-location include tool bugs, lack of documentation, missing features, and support for only a small number of parallel hardware platforms. Proximity to developers increases the likelihood of interaction to aid in overcoming these deficiencies. Tools in this class, however, have passed an initial "usefulness test" by the application community. Hence, the second focus of the center would be to work with users to aggressively test these tools, identify bugs and inadequacies, work with the original tool developers to fix those bugs and add missing features, and (where appropriate) extend the tool to other hardware platforms.

Vendor Cooperation and Standards: Both the promising early prototypes and the more useful, mature tool prototypes should have vendors involved in their evaluation and testing early in the cycle if commercialization is at all an option. Not every tool "vetted" by the center will be a candidate for adoption by one or more vendors; however, many would viewed as sufficiently useful to be of interest to vendors of high-performance parallel systems.

Software Packaging and Support: Finally, tools deemed useful and worthy of dissemination, but not adopted by one or more vendors, must be documented, packaged for installation at remote sites, and, when problems arise, supported, patched, and upgraded. Although this work is mundane, some entity must assume this responsibility, else even good tools will not be used for long periods.

Summary The need for "better" software performance analysis and debugging tools (where better means easier to use, more efficient, better integrated, and more informative) for high-performance parallel systems is a well documented and widely recognized need. The Strategic Implementation Plan [1] of the Committee on Information and Communications of the National Science and Technology Council has noted that "Raising the productivity of the software industry through simplifying toolkits can yield significant dividends in the international marketplace and enable more rapid introduction of hardware advances into affordable production systems." Raising the productivity of applications developers through appropriate, easy-to-use software performance and debugging tools creates a larger market for these affordable production systems. Providing a place where these tools can be effectively tested, evaluated and improved can ensure success in the entire high-performance parallel computing industry.

References

1. Committee on Information and Communications, National Science and Technology Council, "Strategic Implementation Plan: American in the Age of Information," March 1995

2. Koelbel, C., Loveman, D., Schreiber, R., Steele, G, and Zosel, M., The High-Performance FORTRAN Handbook, MIT Press, Cambridge, MA, 1994

3. Messina, P. and Sterling, T. (Eds.), Pasadena Workshop on System Software and Tools for High-Performance Computing, SIAM, January 1992

4. Messina, P. and Sterling, T. (Eds.), "Second Pasadena Workshop on System Software and Tools for HPC Environments," January 1995

5. Pancake, C., "Collaborative Efforts to Develop User-Oriented Parallel Tools," in Debugging and Performance Tuning for Parallel Computer Systems, Simmons, Hayes, Reed, and Brown (Eds), IEEE Computer Society Press, December 1995

6. Simmons, M. , Koskela, R., and Bucher, I. (Eds), Instrumentation for Future Parallel Computing Systems, Addison-Wesley, 1989

7. Simmons, M. L., Hayes, A. H., Brown, J. J., and Reed, D. A., Debugging and Performance Tuning for Parallel Computing Systems, IEEE Computer Society Press, 1995

8. Simmons, M. and Koskela, R. (Eds.), Parallel Computing Systems: Performance Instrumentation and Visualization, Addison-Wesley, 1990

9. Stevens, R. (Ed.), "Workshop and Conference on Grand Challenge Applications and Software Technology," May 1993

Appendix A - Workshop Agenda

Tuesday - 10/15/96	4:00 p.m.	Registration
	6:30 p.m.	Reception
Wednesday - 10/16/96	7:30 a.m.	Registration
	8:45 a.m.	Keynote -- Ken Kennedy, Rice University "Technology Transfer Paths for HPCC Software Tools"
	9:45 a.m.	Dan Reed, University of Illinois "Software Tools: Sin and Redemption"
	10:30 a.m.	Break
	11:00 a.m.	Dennis Gannon, Indiana University "Does Our Model of Parallel Program Performance and Programming Tool Design Scale to Metacomputing?"
	11:45 a.m.	Christopher Kerr, IBM "Use of Software Tools for Application Development"
	12:30 p.m.	Lunch
	2:00 p.m.	Working Group Sessions
	3:30 p.m.	Break
	4:00 p.m.	Working Group Sessions
	6:00 p.m.	Conclusions of Working Group Sessions
	7:00 p.m.	Dinner
	8:30 p.m.	Dessert/Poster Session
Thursday - 10/17/96	8:30 a.m.	Ian Foster, Argonne National Laboratory "Tools for Network-Based Supercomputing: Lessons from the I-WAY Experiment"
	9:15 a.m.	Andrew Grimshaw, University of Virginia "Supporting Diversity and Performance in Wide-Area Metasystems"

	10:00 a.m.	Break
	10:30 a.m.	Working Group Sessions
	12:30 p.m.	Lunch
	2:00 p.m.	Evgenia Smirni, University of Illinois "Parallel I/O: Problems and Solutions"
	2:45 p.m.	Barton P. Miller, University of Wisconsin "An Overview of the State of Parallel Debugging"
	3:30 p.m.	Break
	4:30 p.m.	Working Group Sessions
	6:30 p.m.	Conclusions of Working Group Sessions
	7:00 p.m.	Clambake
Friday - 10/18/96	8:30 a.m.	Working Group Sessions
	10:00 a.m.	Break
	10:30 a.m.	Working Group Reports
	12:30 p.m.	End of Workshop

Appendix B - Working Group Reports

Working Group Reports Overview

The following working group reports were produced by the chairs of each working group in consultation with the members of the respective groups. The six working groups were organized as follows:

- Technical challenges: performance tools
- Management issues for a software tools center
- center software selection and vendor participation
- Intellectual property issues

- center technical Infrastructure
- Technical issues: debugging

Technical Issues - Performance Tools

Working Group Members

Technical Issues - Performance Tools	
John Cerutti	LANL
Joe Durant	SNL
Greg Eisenhauer	Georgia Tech
Peter Eltgroth	LLNL
Edith Epstein	HP
Adam Ferrari	Virginia
Dennis Gannon	Indiana
Al Geist	ORNL
Andrew Grimshaw	Virginia
Michael Heath	Illinois
Marty Itzkowitz	SGI
Karen Karavanic	Wisconsin
Brond Larson	SGI
John May	LLNL
Nick Nystrom	PSC

Doug Pase	IBM
John Reynders	LANL
Diane Rover	Michigan State
Evgenia Smirni	Illinois
Robert Snelick	NIST
Elizabeth Williams	NSA

Introduction Rapid architectural innovation and ever-higher peak hardware performance are two results of the intense vendor competition in high performance computing we see today. While this is good news for the users, it has stretched academic, laboratory, and vendor software tool groups to the limit, forcing them to continually create tools for changing programming models and new hardware environments. This group addresses the current situation in performance tools and outlines challenges and research issues, technology transition remedies, and possible solutions to problems raised.

Questions posed to the Working Group as starting points include:

- What are the most successful current performance tools today and what are their characteristics?
- What performance tools will be needed for emerging programming models and new hardware environments, such as teraflop and petaflop computing and distributed network computing?
- How does one measure the performance accurately across multiple programming models in a single code (object-oriented, data parallel and message passing)?
- How can the technical issues associated with evaluation, testing, interoperability, and leverage best be addressed through the proposed center?
- What relative roles should hardware (e.g. counters, registers, etc.) play in performance tools for future parallel systems such as tera- and peta-flop-capable systems?
- How can the potential data explosion in software statistics gathering be addressed effectively for these large systems?
- How can data mapping and motion issues be correlated effectively with observed (or measured) program performance?

While definitive answers are not offered for most of these, they helped direct the group's discussion. Additionally, the group referenced issues identified by Dan Reed in his talk, "Software Tools: Sin and Redemption," including: emerging opportunities/problems, multiple programming model support, hardware/software support, source code mapping, adaptivity, supporting multiple languages, and the center for Testing and Evaluation (HPSST Task Force).

Finally, the group refined some of the aforementioned issues and brought forth a few additional ones of interest, as follows:

- In the context of opportunities/problems in metacomputing environments, time-shared, interactive environments, and adaptive, dynamic environments, what are new roles for performance tools? For example, will Quality of Service replace FLOPS as a baseline metric?
- Correctness and repeatability of performance measurements is often a concern to users. How can measurements be validated or trusted?
- How can we leverage the existing desktop computing (i.e. commercial) tool base?
- Should we take a broader view of performance tools (e.g., given new metacomputing environments), including not only tools used by end-users to tune programs, but also tools used by application developers (e.g., optimizers, libraries, compilers) and adaptive software systems (e.g., schedulers) to get good performance?
- How should we address end-user resistance to using tools?
- User dissatisfaction may result from non-prescriptive tools. We need to consider what users want to know from tools, i.e., what's the intended purpose in using a tool. As new classes of systems become harder to understand, it may be essential that tools prescribe possible solutions, such as indicating when algorithmic changes are needed.
- What is the level of tool that can be effectively used by end-users who have no knowledge of the field of high-performance program tuning?

Clearly, with respect to tool development, the group is concerned with trends in computing environments, commercial tool technology, and the use of tools.

Preliminary Discussion

The group first considered the question of successful contemporary performance tools for parallel/distributed systems and their characteristics. This of course raised the question of the scope of tools being considered. Without attempting to be comprehensive, we noted a number of traditional tools, commercial and otherwise, that came to mind, such as: ParaGraph, Pablo, Paradyn, XPVM, AIMS, Apprentice and ATExpert (Cray), Workshop (SGI), Puma (HP/Convex), VT (IBM), Prism (TMC), ATOM (DEC), FORGE, Purify, gprof, and timer calls. We extended our list with optimizing compilers, libraries such as PVM, MPI, and ScaLAPACK, languages such as perl, etc., all of which assist the programmer in realizing high performance. In the end, we delineated three broad classes of tools that support: (1) problem identification, e.g., profiling tools; (2) root cause analysis, e.g., trace- and search-based tools; and (3) program development, e.g., programming environments. These tools may provide various types of performance information related to the memory hierarchy (e.g., cache performance), memory utilization, CPU usage, input/output, message passing, synchronization, load balance, processor utilization, source code, etc. This information may be presented at various levels of granularity (e.g., system, node, program, subroutine, basic block, loop, instruction).

The success of these tools is open to debate, or at least the meaning of success is. All are documented, available to users, demonstrated by developers/vendors, and cited in research/product literature. A tool's success is influenced by several factors, including: the tool itself (e.g., quality of its user interface); functionality and applications of the tool; willingness of users; and demands of the application domain. There is no single model of development and deployment leading to success. Moreover, it's hard to pin down the characteristics distinguishing a successful tool. Ideally, a tool should be all of the following: robust, easy to learn (supportive of incremental learning), easy to use (providing quick, basic feedback), portable, configurable, matched to usage idioms, well-supported, flexible (applicable at any time in program development cycle, e.g., without recompilation of program), and accessible (providing readily identifiable functions). Ultimately, the key characteristic of a successful performance tool is that information provided by the tool is meaningful and useful for improving program performance.

A useful reference relevant to this discussion is "Techniques for Performance Measurement of Parallel Programs," by J. Hollingsworth, B. Miller, and J. Lumpp, in Parallel Computers: Theory and Practice (edited by T. Casavant, P. Tvrdik, and F. Plasil, IEEE CS Press, 1996), which summarizes problems associated with developing performance measurement tools and describes some of the systems that have been built to deal with these problems.

Challenges in Tool Development and Application

The group identified a number of challenges faced by the performance tool community looking ahead to future needs. These represent hard problems that are currently without solution. We summarized three areas for these challenges:

- Heterogeneity
- Hierarchy
- Scale

All aspects of a parallel/distributed application are becoming increasingly heterogeneous. The application itself is not only compute-intensive but also data-intensive, involving scientific instruments, databases, etc. The architecture upon which it executes has potentially diverse resources in the systems used, their nodes, the networks connecting them, and so on. The computation layer and the network layer exhibit intricate relationships and cannot be viewed separately. The heterogeneity in application and architecture propagates into the programming languages, environments, and models. This heterogeneity often results in more dynamic, less predictable conditions so that adaptability--another challenge--is essential. Consequently, validity of measurements is even more difficult to determine.

Hierarchy is an important element in system design and analysis and is often exploited to manage complexity. It is present throughout high-performance computing, in systems, networks, memory, algorithms, programming abstractions, etc. However, to take advantage of its power, we first need to expose it from a performance perspective. This remains a challenge for performance tools.

Finally, solving grand problems involves ever-grand scales. The scales of the following attributes continue to grow: number of processors, size of application, length of run, number of resources to be controlled/managed (e.g., network, cache), and amount of concurrency, among others. The amount of concurrency is related in part to the finer granularity of parallelism, such as threads and instruction-level parallelism. The volume of performance data collected and the scale of the information are vast. Effectively dealing with these is another significant challenge to performance tools. Moreover, tools should offer insight into scalability and highlight behavior impacted by relevant scaling factors. This is only becoming more important as users have access to varying processor counts in multiprocessor workstations as well as in network-based supercomputers.

In short, with heterogeneity, hierarchy, and scale, the end goal of optimization of program performance becomes more complicated (or, as one participant stated, "NP-really-hard").

Selected Technical Issues

The most important technical issues agreed upon by the group address emerging computing environments and the challenges outlined above. They are summarized briefly:

- Selective presentation given data explosions
- Memory utilization and exploiting NUMAness
- Tuning across multiple executions, systems, and networks
- Mixed programming models: message passing, object-oriented, data parallel, threads, etc.
- Mixed languages and paradigms: C, Fortran, HPC++, etc.
- Adaptability to dynamic conditions
- Localized vs. system-wide performance analysis
- Accuracy and validation of results
- Tuning of scheduling algorithms
- Real-time control of data collection: volume, level, type, etc.

Some of these have special relevance to metacomputing environments and high-end shared-memory systems. In a metacomputing environment, traditional tuning techniques for performance optimization, e.g., per-node component optimization, will continue to be used; however, new tuning techniques are essential. The objective of tuning focuses on high quality of service or best average performance. It may also involve non-performance oriented metrics, such as cost. Balancing of per-node performance with system-wide performance must take into account considerable variability in resource selection, availability, and obtained performance. Tuning across multiple executions may be needed for useful results. More automated tuning is likely required due to increased complexity. For example, tools may be required to adjust resource usage on the fly, dynamically adapting to changes in the environment (e.g., new users, faults, load fluctuations, etc.). Tuning scheduling algorithms may have greater impact than tuning application code. Issues include resource requirements for scheduling and other system functions, such as network management, and performance estimation.

In a shared-memory system, control of memory allocation, including reallocation, is important to enforce locality. In general, NUMA-related features of the architecture must be exploited to achieve high performance. With clusters of symmetric multiprocessors, tools need to expose the hierarchy and both message passing and shared memory models.

Issues involving users and their views of performance continue to be a concern. Tools must support and incorporate domain-specific performance information in metrics and displays so that information gathered and presented is better matched to the programming model seen by the user. Minimally, tools needs to map data back to the source code. Users tend to prefer using a single programming model, for example, adopting a message-passing model even on shared-memory machines (allowing portability as well as efficient implementation); and to resist new machines unless critical for performance. Thus, tools must handle information about an application in terms of the programming model and with respect to the machine. However, different users have different interests: an application programmer may be interested in viewing data at a high level of abstraction, while a toolkit builder may need low level data. But users may also be interested in the costs to program execution of mapping between models. In general, providing feedback to users on implementation aspects that may cause performance problems is of growing importance (for example, if some procedure or parameter may cause many small messages, which is known to be costly).

Many of these technical issues, or related ones, have already been the subject of discussion among tool developers and users. While progress has been made, rapid changes in hardware and architecture result in a moving target.

---

center Commentary The group explored activities and issues associated with the proposed center for testing and evaluation of software tools. Possible activities for a center and the group's assessment of the level of each activity are listed:

- Evaluate research prototypes: High
- Develop enhanced prototypes, deploy and maintain prototypes: High
- Test/validate prototypes: High
- Promote commercialization: Low
- Promote standards: Low-Moderate
- Facilitate interoperability: Low-Moderate
- Promote user involvement: High

A number of questions were raised, as summarized in the following issues. The group expressed concern about the level of funding to the center and the effect on funding to external (non-center) research projects. Additionally, the mode of collaboration, interaction, and transition with external projects needs to be addressed. In particular, the process of selecting tools, i.e. transitioning, must be specified, including the scale of projects/tools to be undertaken. The roles of the stakeholders are not yet clear. Why would researchers hand over projects to a center? Why would agencies or labs hand over critical software? If it's non-critical software, then what is the reason for transitioning and investing in it? On the other hand, to what extent will vendors be willing to participate in the center, for example, adhering to standards developed by the center? How can the center ensure vendor support for tools that need specific hardware support (that must be provided by the vendor)? Another concern is staff quality and diversity of skills. The center will need to attract well-qualified technical staff having a potentially wide variety of skills. Is that possible if the reality or perception is that they are working on mundane aspects of others' projects? Finally, the group noted a potential conflict of interest in the role of the center to both develop and test prototypes.

In further considering the center, we assumed it would not be the dominant means for tool development. In other words, development would continue to occur by various research groups. For appropriate projects undertaken by the center, it will enhance and maintain tools. However, the center will have resources sufficient to address only a subset of the technical issues. Given this situation, potential positive outcomes of the center include: development of more robust tools for multiple platforms; greater availability of tools; availability of canonical benchmarks for tool testing and evaluation; and education. Several alternative strategies instead of setting up a center were cited:

- Change procurement policy so software tools are explicit in RFPs; then vendors will transition tools.
- Recognize the intrinsic value of software tool development separate from application development (as Infrastructure across applications).
- Consider other models, such as the Digital Libraries Project (pilot implementations at multiple sites).

The group took an objective albeit selective look at the proposed center, as time and assumptions permitted. We sought to identify both positive and negative aspects, particularly from a technical viewpoint. Whereas many questions remain to be answered, the exercise--a forum investigating new ideas and strategies--was worth the investment. There are many challenging technical issues facing the performance tool community.

Acknowledgements Special appreciation is given to Greg Eisenhauer and Adam Ferrari for serving as scribes for the sessions and to Al Geist and Mike Heath for assistance in extracting highlights for the presentation.

Management Issues for Proposed Software Tools center

- Working Group Members
- Introduction
- Answers to Management Issues
- Additional Questions and Basic Assumptions
- Summary and Additional Comments

Working Group Members

Management Issues for Proposed Software Tools center	
Ann H. Hayes	LANL
Kathleen P. Hirons	LANL
Kay Howell	DOD
Alan Mink	NIST
Richard Oliver	NMSU
James C. T. Pool, Chair	Caltech
David Rich	Dolphinics
Nahid Sidki	DOD
Paul H. Smith	DOE
Rick Stevens	ANL

Tammy Welcome	NERSC
Mary Zosel	LLNL

Introduction

The charter of Working Group 2, Management Issues for Proposed Software Tools center, was formulated as the following set of questions:

- What is the mission of the center?
- What are the goals and objectives of the center?
- Define the scope of the proposed center.
- Envision the management structure of this organization. How large and what skills must its staff possess?
- Should the center be centralized? Distributed? A virtual center?
- How should such a center be organized for each of the above models?
- Shall center personnel be permanent? Temporary? Rotating?
- What infrastructure should be in place to facilitate interactions between vendors, developers, and users?
- What should be the model for achieving these interactions and how should funding be allocated (percentage, method) for these collaborations?
- What sort of oversight process should be put in place at the center?
- What lessons can be gleaned from the past experience (e.g., the NSF supercomputer centers and S&T centers)?

In this report, we first answer these questions (in the order the questions were considered by the Working Group). We then summarize: a) additional issues raised during the Working Group's initial deliberations about these questions and b) the assumptions developed during these initial deliberations and employed to prepare the answers to the questions.

Answers to Management Issues

Question 1: What is the mission of the center?

The mission of the proposed center should be to ensure application developers have access to major software tools responsive to their requirements for developing applications on highly parallel HPC systems by stimulating interactions among application developers and software tool developers, including the computer science research community, independent software vendors (ISVs), and system vendors.

Question 2: What are the goals and objectives of the center?

The goals and objectives of the proposed center should be to:

- collect, collate (to examine and compare carefully in order to note points of disagreement), and disseminate users' requirements for tools;
- stimulate standardization to ensure interoperability among tools and portability of tools;
- establish methodology and test suites to evaluate tools;
- evaluate and, if appropriate, enhance prototypes of major software tools;

- ensure enhanced prototypes are available to application developers as high quality tools on the principal highly parallel HPC systems, and
- support these tools through their useful life period or until alternative support exists.

Question 3: Define the scope of the proposed center?

The proposed center should accept prototypes for major software tools applicable to highly parallel HPC systems, that is, tools with minimal potential for commercialization due to limited market, from application developers and tool developers (including ISVs and system vendors under appropriate licensing agreements that ensure availability on the principal highly parallel HPC systems) to evaluate and, as appropriate, to enhance to high quality tools primarily via partners (see "CRPC Model" below) and/or mini-consortia (see "CNRI Model" below).

Questions 5 & 6: Should the center be Centralized? Distributed? A virtual center? How should such a center be organized for each of these models?

The center should incorporate the best features of the center for Research on Parallel Computation and the Corporation for National Research Initiatives. The center for Research on Parallel Computation (CRPC) is one of the National Science Foundation's 25 Science and Technology centers. The CRPC was established in 1989 to make massively parallel computing systems as usable as conventional supercomputing systems are today. The CRPC is a consortium that includes seven core sites and six affiliated sites; therefore, the term "center" implies not a single facility, but a nation-wide consortium of more than 400 researchers, support staff, and graduate students.

The Corporation for National Research Initiatives (CNRI) is a non-profit research and development organization formed in 1986 to help focus U.S. strengths in information processing technology. Working with industry, government, and academia, CNRI is engaged in scientific research on the design of experimental infrastructure which can improve the country's long-range scientific and engineering productivity. CNRI conceives and leads multi-party collaborative research efforts involving U.S. Government, business and academic institutions. Among its numerous projects, CNRI organized and managed a national gigabit network testbed project with the support of the National Science Foundation (NSF) and the Advanced Research Projects Agency (ARPA), involvement of key universities and U.S. computer and telecommunications companies.

A "CRPC Model" for the proposed center would emphasize an initial set of primary partners (subsequently augmented with secondary participants as deemed necessary) to evaluate and then to enhance selected prototypes. A "CNRI Model" for the proposed center would emphasize an initial small core staff responsible for establishing mini-consortia to evaluate, enhance, and maintain prototypes in specific areas. Thus, a fundamental difference between these two models is their initial formation -- the "CRPC Model" involves a pre-defined set of partners from the beginning, while the "CNRI Model" has only an initial core staff.

Advantages of the "CRPC Model" include:

- Coherent, stable team of primary partners
- Technical staff with "corporate memory"
- Team available to initiate new ventures without startup delay
- Executive Committee management

- Reallocation of funds to respond to unanticipated opportunities
- Continuing assessment of progress and directions by key participants
- Coordination among partners to ensure interoperability
- More responsive to smaller tools
- Experts available to quickly assess ideas

Advantages of the "CNRI Model" include:

- Open structure
- Evolving set of participants
- Easier to terminate projects and initiate new projects
- Mini-consortia can be formed to ensure involving all segments of HPC community interested in a specific tool
- Flexibility
- Choice of projects and their arrangements
- Subcontracts including intellectual property
- Close interactions with agency program managers
- Selection of projects consistent with agency programs

These distinctions are, of course, not totally exclusive; moreover, both models would support:

- organizing workshops and forums to determine user requirements, interfaces among tools, and system interfaces, and
- providing basic technical services, e.g., generate test suites.

In both models, each project team should include:

- Application developers (academia, centers, laboratories),
- Tool developers (academia, laboratories, ISVs, vendors), and
- Software engineers (laboratories, ISVs, vendors).

Moreover, the project team should include representatives of the developers of the prototype to be enhanced.

Question 4: Envision the management structure of this organization. How large a staff? What skills must its staff possess?

The proposed center should include a core staff to manage interactions with users, developers, ISVs, vendors, and agencies. Under the "CNRI Model", this staff would, for example:

- generate statements of interest in specific tool areas,
- review resulting white papers and proposals,
- negotiate and monitor subcontracts to mini-consortia,
- review mini-consortia's technical progress, and
- provide limited technical support, e.g., generate test suites and participate in review process.

Under the "CRPC Model", the core staff would interact with the partners rather than the mini-consortia and this core staff would be augmented by additional technical staff at the lead site.

Question 7: Shall center personnel be Permanent? Temporary? Rotating?

The proposed center should include a "permanent" management and administrative staff, leverage extant staff and facilities, and include small rotating technical staff. This staff should be located at an organization with appropriate supporting infrastructure ranging from a procurement office to employee benefits (and appropriate computing resources which, however, is a topic included in the charter of another Working Group).

Questions 8 & 9: What infrastructure should be in place to facilitate interactions between vendors, developers, and users? What should be the model for achieving these interactions? How should funding be allocated (percentage, method) for these collaborations?

The "CRPC Model" and "CNRI Model" discussed above provide excellent examples of successful infrastructures. Under either model or a blending of the two models, the center should initiate test and evaluation activities in a small number of areas consistent with perceived user requirements. The center should then initiate one or possibly two primary long term (e.g., three year) projects per year until it has achieved a steady state consistent with the available funding. It should organize corresponding major forums and workshops and, as appropriate, initiate perhaps two smaller short term projects per year.

Question 10: What sort of oversight process should be put in place at the center?

The proposed center should have:

- an independent user group with members representing users of the principal highly parallel systems and a cross-section of application disciplines,
- a small executive committee charged with the responsibility of managing the center,
- a technical committee (application developers, tool developers, and tool providers involved in projects),
- an external (technical) advisory committee (application developers, tool developers, and tool providers), and
- an inter-agency (policy) advisory committee (e.g., responsible agency program managers).

Question 11: What lessons can be gleaned from the past experience (e.g., NSF supercomputer centers and S&T centers)?

The answers to the questions have been built from experience with CRPC and CNRI. The experience gained during the preparation of proposals for the NSF Partnerships for Advanced Computing Infrastructure Program indicates potential opportunities for cooperation and possibly direct involvement with the successors to the NSF Supercomputing centers. Likewise, there are potential opportunities for cooperation with DOE laboratories, DOD centers including the DOD Modernization Program, and the NASA research centers. These opportunities include, in particular, mechanisms for involving application developers.

Additional Questions and Basic Assumptions

The above answers were based an a set of assumptions adopted during the preliminary deliberations of the Working Group. These assumptions resulted from an initial cursory review of the questions, an activity which raised additional questions and issues, including the following:

- Why a center? What does it contribute that current programs do not provide?
- How do you define "tools," "users"?
- What drives the center: users' requirements, developers' ideas, and/or ISV/vendors' requirements?
- How do you communicate users'requirements to developers (different than user involvement in testing and evaluation)?
- How do you ensure "real" feedback from users?
- How do you select and support projects without becoming a "funding agency"?
- How do you ensure center helps ISVs and vendors, rather than competes?
- Who are the center's customers? Who pays the bill for tools?
- What is anticipated level of annual funding?
- What are the center's deliverables? Who promotes these deliverables?
- How do you ensure interoperability of tools?
- How do you sustain center when initial enthusiasm declines?

The result of these initial deliberations was a set of assumptions, including:

- The raison d'etre for the proposed center is providing tools to make an impact on application development; therefore, the center should focus on major software tools for highly parallel HPC systems -- like hardware, these tools have minimal potential for commercial viability;
- Application developers' requirements should be the driving force for the center;
- while the customer for tools is the application developer, the cost is incurred by agency application programs
- distinct applications have the same needs for tools which implies multi-agency funding

- probably less than ten candidate prototypes for ready for enhancement today
- implies pursuing a few large multi-year projects
- sources for prototypes
- basic and applied computer science research programs (academia, laboratories)
- major mission programs (e.g., DOE/DP ASCI, DOD HPC MOD/PET, NSF PACI, NASA GCs)
- the center output must match/exceed quality of a commercial package
- Implies continuing support for successes
- the center will leverage existing HPC activities
- center can benefit from Ptools and NHSE input from users
- center may agument NHSE parallel tool repository

Summary and Additional Comments

The Working Group conducted its deliberations about the management issues for the proposed center without debating whether or not the center should be formed.

Since the mission of the proposed center is development rather than research, the Working Group favored the "CNRI Model". However, the Working Group recognized that a compromise between the two models through an assimilation of selected features of the "CNRI Model" into the "CRPC Model" is a possible alternative. A "modified CRPC Model" might include, for example, strengthening the roles of the Director and the Executive Committee to manage development activities, starting with a smaller set of partners, and adding partners selectively as the areas of software tools to be evaluated and enhanced are more completely defined.

The Working Group questioned how to sustain the proposed center when the initial enthusiasm declines? It concluded that the user community through, for example, a user group could be the ultimate advocate for sustaining the center. The success of the center would then be judged by the users of the software tools it delivered, thus, ensuring that user requirements drive the activities of the center.

Software Selection and Vendor Involvement

- **Working Group Members**
- **Introduction**

Working Group Members

Software Selection and Vendor Involvement	
Jim McGraw (Chair)	LLNL
T. Adams	LANL
Robert Dilly	IBM
C. Kerr	IBM
James Kohl	ORNL
Olaf Lubeck	LANL
Allen Malony	Oregon
Allan Porterfield	Tera
S. Sekiguchi	Electrotechnical Laboratory
Margaret Simmons	NCO
Pat Teller	Texas, El Paso

Introduction

A Software Tools Testing and Evaluation center (STTEC) must facilitate the use of high quality and innovative tools by the developers of scalable, parallel scientific applications. Such tools are not available. Tools developed by researchers in the academic sector, although innovative, usually do not evolve into usable tools that could be adopted as or integrated into products by vendors. Tools developed as part of commercial efforts usually lack the innovation and investment necessary to solve the hard problems related to application development. This working group was asked to focus on two key questions related to the structure of a STTEC, which we will call the center: (1) What selection criteria should be used to identify the software to be developed at the center? (2) How might vendors participate in the center's activities and benefit from the center's accomplishments?

The discussions and conclusions from this working group covered a broad range of possible mission statements for a center and was influenced by the desired degree of vendor involvement. A number of important points surfaced regarding difficulties that various types of vendors might encounter in taking advantage of the testing, evaluation, and hardening efforts that the center might undertake. It appears that while a center should encourage the participation of vendors as much as possible, the primary customers should probably be the application developers that use high-end systems.

Using this focus, the working group elaborated on a possible mission/charter for the center. This charter included evaluation, testing, and hardening activities for various software tools. Based on these activities, the group developed selection criteria for choosing software that a center would handle. These criteria included factors related to the state of the tool (current and proposed), the viability of the technical plans for advancing the tool, the resources needed to carry out the plan, and various requirements upon the center itself. In addition to these criteria, this group recognized that the Working Group on Intellectual Property Issues raised some significant points that needed to be addressed as part of the selection criteria (which are not replicated here).

The remainder of this working group report contains four sections. Section 1 summarizes the key issues and conclusions regarding the potential for vendor involvement in a center. Section 2 gives a detailed description of the assumptions made about the structure and charter of a center. Section 3 contains the description of the selection criteria for transitioning software tools into and out of the center's responsibility. Section 4 summarizes the key conclusions of the group's deliberations.

Vendor Involvement Comments and Analysis

Any long-term solution to the inadequacy of software tools for high performance computing (HPC) must involve vendors because only they can provide the necessary infrastructure support and migration of these tools to new platforms. In this context, "vendor" refers to a computer manufacturer or independent software vendor. If a center is proposed to help solve the software tool inadequacy, then seeking vendors' acceptance and use of the center is critical. Not surprisingly, vendors have a diversity of positions and concerns regarding what a center might contribute and what type of vendor involvement would be most beneficial and desired. This section highlights the Working Group's discussion of vendors' perspective on software tools for HPC, evaluation vs. hardening options for a center charter, and the potential roles vendors could play. Conclusions arising from these discussions make it clear that vendor involvement is a challenging problem that will require significant care and attention in any center proposal.

For hardware vendors, revenue comes from selling hardware. Software tools are only critical when they directly affect the purchase of hardware. In such an environment, the center must make the transfer of technology to vendors very simple. Quantifying exactly what users need will be key. Once identified, passing these solutions to vendors must be straightforward. Sometimes vendors will be willing to take source code supplied by the center directly. Other times the basic idea can be incorporated into an existing tool (using all, some or none of the original source). Some vendors will like an idea but will be unable to use the supplied code and will need to rewrite the code to meet their internal standards. These standards cover issues such as internationalization, service tools, compatibility with vendors' related products and even look and feel. In all these cases, the center staff will need to supply expertise to make sure the transfer of technology succeeds. The assistance will range from consulting on the tool design to supplying users for evaluating the vendor's implementation.

The test and evaluation functions of the center were perceived to be most universally attractive to vendors, particularly if the testing were to provide favorable results that could be viewed as an "endorsement" of the marketability of the software. In addition, any feedback from the evaluation could provide independent information for planning and further commercial investment. However, if the software is proprietary, this test and evaluation function and implied endorsement likely would be incompatible with the nature of a center as an independent, government-funded organization. Thus, the center might only accept non-proprietary tools. Even so, companies may be willing to provide tools if improvements (e.g., making them interoperable over many platforms and more robust) would enhance the value of their hardware. Universities might also be willing to provide software that they would not otherwise commercialize and for which they seek broader acceptance and use.

The center could begin with "donated" software tools meeting certain initial evaluation criteria. The center's test and evaluation functions would identify both the potential value of the tool and what kinds of improvements would be needed to transform it from a research state into a commercial prototype. The tool originator and vendors would benefit from the information obtained during the test and evaluation. If the center would then proceed to further develop the tool, the entire community would benefit from having a more robust, better documented, commercial prototype. This prototype could then be distributed through NHSE and be commercialized by any interested vendor.

A significant group of vendors not mentioned so far, are Independent Software Vendors (ISV). ISVs are, if anything, a more delicate problem. Such tool vendors' revenue is often closely tied to the uniqueness of their products and the breadth of platforms on which they are supported. The center must take particular care not to "compete" with their products. At the same time, ISVs have experience at porting between machines, which could be an excellent resource for the center. If a center could find ways to work with ISVs to help "harden" programs, it would be beneficial (e.g., hiring an ISV to consult on a project or contracting with an ISV to perform the hardening and documentation). These types of arrangements lead to a natural migration path out of a center and to the ISV to create a supported product. By finding and working with ISVs, a center might be able to satisfy the trickiest legal problem (not overlapping ISVs) while providing an exit process for successful programs.

The big question in joint development between a center and ISVs will be ownership. This issue needs to be resolved early in the operation of any successful center. A single policy on ownership needs to be adopted and very few, if any exceptions, need to be made to it. Consider the life-cycle of a typical tool which starts in a university setting with many students contributing which is then transferred to the center. The center applies its staff to evaluate, harden and test with input from some set of users to enhance the tool. By the time a tool is ready for commercial adoption, many individuals will have contributed to it. Corporate attorneys will seek legal assurance that the technology from the center can be used in a commercial product without becoming involved in litigation over ownership. For a more comprehensive discussion of these issues, refer to the Working Group on Intellectual Property Issues.

A second question will be whether a vendor (ISV or hardware) can supply code to a center for evaluation. The group considered it unlikely that any vendor would submit any software "critical" to the company's success. However, the general feeling (with dissent) was that vendors should be able to submit potential codes just like any research center, applying the same qualifications and ownership rights.

Assumed center

Given all of the options and uncertainty to arise out of the vendor involvement discussions, the working group decided it had to focus on a particular vision of a center and its charter. Such a focus suggests one detailed approach to the direction a center might take and it provides a

Structure more concrete platform for discussions about selection criteria for tools entering and leaving the center's responsibility. This section describes the working group's vision for a Software Tools Testing and Evaluation center. Figure 1 shows a simple model of the participants and expected products. In this model, the primary customers of the center are the people who develop applications for HPC systems. The center's activities would include testing, evaluating, hardening, documenting, and/or consulting for selected software tools. An Evaluation Board would provide direction for handling proposals for the center's involvement in any software tools. The working group believes that such a center might naturally fill a gap between the current Parallel Tools Consortium and the National HPCC Software Exchange.

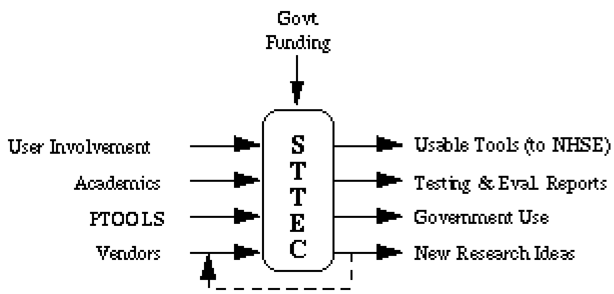

Figure 1: Working Group Model of Participation and Products for a center.

The working group concluded that the primary customer focus for the envisioned center needed to be the HPC application developers. In this model, they will present requests for tools or tool improvements to the center and will offer their time as volunteers to assist in tool evaluation and testing. The rationale for putting application developers first reinforces the point that they are the people who need the products of the center; it is their time and energy that is being leveraged to produce more effective software. Their input on the selection of software to be handled by the center and their recommendations on how the center's activities can enhance the use of the software will be of critical importance. The working group felt vendors could not be viewed as the primary customer of the center, due to the risk involved in investing in speculative tools. If HPC computer systems are viewed as a relatively low volume market, then the software tools that support those platforms must capture a large market share to be profitable.

The Federal Government is an important "customer" of the center because it is expected to provide the bulk of the funding needed to make the center a reality and because many of its critical missions can only be met through effective use of HPC systems. As such, the government tie adds some important constraints (e.g., public access to results and non-competition with the private sector) and responsibilities (e.g., an effective portfolio of software tool investments that span the needs of the entire HPC community).

Proposals to the center could include the enhancement of an existing prototype tool, the development of a proposed tool, and the testing and evaluation of existing prototype tools. Potential sources of proposals span the entire community. Software researchers (from the academic, government, and commercial sectors) often develop innovative prototype tools. Such tools might provide new capabilities or use new solution techniques. However, due to funding constraints or other practical reasons, these tools are not hardened products. Proposals from these researchers would further efforts to make the tools usable by the user community. In contrast, members of the user community might propose desirable tools that currently do not exist at all. Vendors could propose testing and evaluation of their prototype tools or tool "kernels." They could also propose to contribute non-proprietary tools that could be made available to the community on an "as is" basis. Of course, ideas or tool elements produced by the PTOOLS consortium can be incorporated into a proposal to the center.

Members of all three potential sources of proposals can benefit from the efforts of the center. The research of software researchers can be enhanced by their building on top of hardened, state-of-the-art software instead of research prototypes. The user community as a whole can access state-of-the-art software development tools. Vendors can have prototype tools tested and evaluated by the center, thus, facilitating the software development process, and can direct their user community to center tools.

Figure 2 illustrates the envisioned center activities and work flow. Different tools will have different activity paths through the center. An Evaluation Board will manage the path of each tool based on proposals received for consideration. To facilitate the job of the Evaluation Board, any existing tool that is part of a proposal will receive an initial test and evaluation by center staff. This evaluation will be made public at some point in time. The time at which the evaluation will be made public may depend upon vendor considerations. In this way, vendors can propose that the center periodically review a tool throughout its development, without breaching confidentiality before the tool is released. The center will create proper documentation for tools that it "hardens" to meet real-world usage and for tools that it develops in their entirety. In either case, the center will be responsible for supporting these tools for some reasonable time period and will provide related additional documentation, maintenance, and consulting as needed. However at some point, every tool is expected to exit the center. At that time, the center relinquishes all responsibility and ownership for the tool.

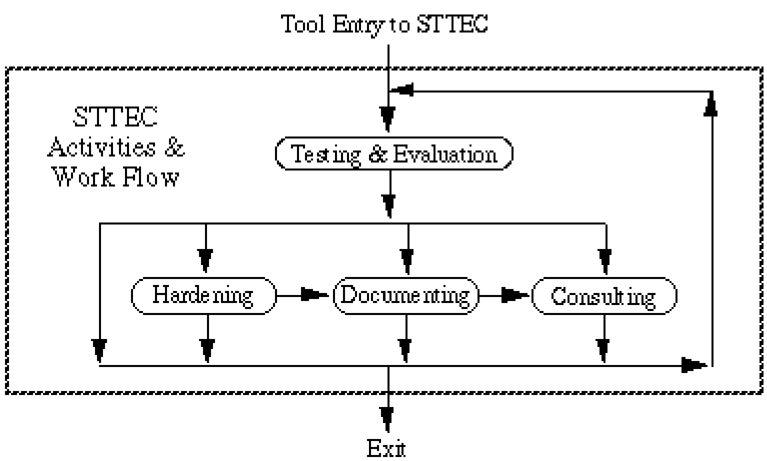

Figure 2. center Activities and Workflow Structure

An initial solicitation for proposals would be generated to initiate the center's activities. After this initial solicitation, proposals could be submitted anytime. All proposals would be evaluated by the center's Evaluation Board (which we will call the Board) on a regular basis. As mentioned above, a proposal could be: (1) a request for test and evaluation only, (2) a plan for "hardening" a prototype tool by the center, or (3) a plan for developing a new tool from scratch by the center. For each proposal, the Board would assess the anticipated costs and benefits, as well as the availability of required resources. Given this information, the Board would rank proposals according to a defined selection criteria, which is discussed in the next section.

The center's Evaluation Board would be comprised of a mixture of researchers, vendors, users, and center staff. The board, as a whole, would be required to have expertise in the field of high-performance computing, and sufficient knowledge and experience to be able to evaluate and estimate the cost of center activities that would be associated with implementing software tool proposals. Researchers will be able to evaluate the difficulty associated with enhancing or developing a tool. Vendors will be able to identify tools that would be useful to their user base or applicable to upcoming architectures. Users will be able to identify tools that would be adopted by the user community. center staff will be able assess the cost of processing (be it test, evaluation, enhancement, development, or other support). Board members should serve on a staggered, rotating basis. This approach would insure continuity of decisions and participation of different vendors, universities, and government laboratories and agencies. Travel expenses for academic Board members would be reimbursed.

The working group's vision of the center fits comfortably with other existing organizations that are actively involved with software tools efforts, including the Parallel Tools (PTOOLS) Consortium and the National HPCC Software Exchange (NHSE). The NHSE acts as a distribution service for software, documents, data, and information of interest to the high performance and parallel computing community. As such, it promotes software sharing and reuse within and across HPCC agencies and facilitates the development of discipline-oriented software and document repositories. The NHSE is the logical place to distribute software "produced" by the center.

The mission of the PTOOLS Consortium is to take a leadership role in defining, developing, and promoting parallel tools that meet the specific requirements of users who develop scalable applications on a variety of platforms. PTOOLS focuses on new tools, guiding the development, testing, and evaluation of them from inception through reference implementation. All tools must meet a demonstrable needs in the user community and the PTOOLS processes involve users throughout the development. Funding to support the tool development work is the responsibility the proposer of a PTOOLS effort. The Consortium provides a forum for potential tool users to guide the tool implementors toward products that will have the features and portability to have broad acceptance in the community.

The working group viewed the activities of PTOOLS and the center as complimentary. Both share a common mission and stress the importance of user involvement toward the development of hardened tools. The primary differences focus on the types of activities to be undertaken and the means for financially supporting them. The center is expected to have immediate resources (money and personnel) to dedicate to carrying out its activities and harden successful research prototypes developed elsewhere. The center can also test and evaluate software developed elsewhere. Hopefully, the combination of the two approaches to tool development will increase dramatically the availability of high-quality tools and expand their use in the community.

Selection Criteria

Having come to agreement on the structure and charter for a proposed center, the Working Group encountered little difficulty identifying appropriate selection criteria for software to be brought into the center. The criteria seemed to divide naturally into five categories: current state of the tool, proposed state of the tool, the plan for advancing the tool, resource considerations, and "center" considerations. All of the criteria include a high level of subjectivity. This section elaborates on the working group's vision and understanding of these criteria and how they could be effectively applied in practice. Clearly, a tool may not be able to "score" well in all of the criteria, however any tool submission should address these issues. The Evaluation Board for the center must be responsible for carefully weighting these different criteria to best support the HPC user community's needs.

The state of a tool upon entry into the center determines the fundamental value of the tool as it currently stands and indicates the potential for useful center improvements. Likewise, the proposed state of the tool provides an evaluation of whether it would be a worthwhile endeavor for the center to work on the tool, or whether the end result itself would be useful. The qualities evaluated for the (current and planned) state of the tool include:

- User Support -- the tool must have an existing user base or users must be desirous of such a tool. In addition, some experimental user community must be able to consult while the tool is under center supervision. This aspect emphasizes immediate impact for the development of HPC applications.
- Tool Value -- includes analyzing the relevance, timeliness and viability of the tool, as well as how far-reaching the impact of the tool may be on the user community. A tool should have a wide range of impact within the HPC community at large, or at least a high impact within some specific user community, to be accepted by the center. This aspect emphasizes more of a long-range potential of the tool.
- "Plug-in" Compatibility -- it must be possible to integrate the tool's functionality with other complete tools. The tool should also be extensible. If not in the current state of the tool, these characteristics almost certainly need to be included in the proposed state.
- Interoperability -- a tool should support sufficiently general application and user interfaces so as to integrate well and interoperate with a variety of commercially available development environments and hardware.

center activities are meant to realize state-of-the-art hardened tools and make them available to the user community. To attain this goal in a timely manner, center activities associated with any one tool must be limited to a reasonable number of person-hours. The plan for handling each tool must be clearly delineated and meet the following selection criteria.

- Well-defined Activities -- the activities associated with the processing of a tool (be it test and evaluation, enhancement or development) must be clearly defined and attainable. Thus, it is of utmost importance that the enhancement or development of a tool not require any research activities.
- User Involvement -- periodic test and evaluation during the development of a tool by the user community is imperative.
- Measurable Milestones -- attainable milestones, as well as related test and evaluation procedures, must be defined. (Such test and evaluation procedures may trigger the tool's exit from the center's work flow, if milestones are not met.)

The center makes a resource commitment to any tool that it accepts. This investment is made in expectation of returns on the investment accrued by later users of the tool. Hence, the type and level of resource requirement for the tool project within the center should be considered as part of the selection criteria. Resource needs include the "expenditures" of personnel (center staff) over the lifetime of the project and the money needed to support their working environment. The tool might also require certain hardware and software infrastructure that the center may need to buy and then maintain. Clearly, some tool projects will be larger than others, requiring greater assignment of resources for their successful completion. It is therefore important for the center not only to have a clear sense of resource needs as they relate to the project plan (different project stages may have different needs), but also a method to account for the cost of those resources with respect to the center's operating environment.

Interestingly, the working group considered the possibility of a tool project being proposed with supplemented resources. Resources brought in with a project could come from several places and take various forms. For instance, an interest group of industry and/or user participants might decide that a tool is important enough to sponsor its testing and evaluation in the center through money, the donation of machines, or the allocation of personnel. Such direct support goes to offsetting the costs of the tool project. It is also conceivable that there may be other forms of indirect support proposed such as granting access temporarily to computing systems, providing consultant services, certain "after tool completion" service, etc. Any form of supplemented resources should be weighed in consideration of tool project acceptance.

The selection criteria identified so far focus on attributes of each individual proposal. In addition, the center must consider broader criteria that examine each proposal in the context of the overall effectiveness of the center and its ability to serve all of its customers. It makes no sense for the center to support five different flavors of the same type of tool, even if they all have their own important user communities. Through careful selection criteria, the center can develop a strong "portfolio" of tools that covers the spectrum from writing code to tuning performance. The center also needs to consider each decision in the light of how it will affect the long-term survivability of the center itself. For the short-term, the center will certainly need some quick wins to show it can deliver. However, for the long-run, the center must take on the hard problems and show some big wins. Such strategic planning must be part of the selection criteria.

Probably one of the trickiest issues for the selection criteria is avoiding competition with industry. If a computer vendor or ISV decides to develop a particular tool that is closely related to a tool proposed to the center (or worse, already under some level of hardening by the center) tough decisions must be made to avoid running afoul of federal law. This situation likely requires the center to be very public about projects even at the proposal stage. Detecting potential conflicts early may avoid having to cancel a project already in progress and may actually support the goal of helping move new technology into commercial products.

Conclusions

The discussions in this working group led to three specific types of conclusions. Much of the actual time in the meetings centered on the basic question of what kind of role and charter for a Software Tools Testing and Evaluation center might make sense and prove most valuable. Significant skepticism was expressed about the value likely to be returned for expected high cost of such a center. By the end of discussions, some of these concerns appeared to diminish; however, the diversity of visions for a center produced by the different groups demonstrates the need for a more refined and carefully defined vision for such a center. This working group's first conclusion affirms the needs for this more precisely stated vision and charter. Based on its deliberations, the group favors a broad charter that includes testing, evaluation, hardening, documenting, and consulting (where specific decisions must be made as to which of these activities is applied to each tool in the center).

Vendor involvement in the center turned out to be a very challenging problem. Ideally, one would hope that tools produced by the center would be easily adopted by all of the vendors, based on the center's ability to move valuable research ideas to hardened tools that everyone would want to use. In reality, vendors cannot be viewed as a homogeneous group. They have different policies, procedures, and driving forces that significantly influence their desire and ability to support this ideal. This group's second conclusion is that vendor "involvement" for the center needs to be encouraged throughout the process, but that expectations for results need to be framed carefully. While vendors can and should participate, they should not be viewed as the primary customer of the center. Commercial success of tools handled by the center should not be the metric for project success. Instead, the group recommends that HPC application developers be the primary customers of the center and their use of the tools be the key metric for success. Moreover, if a tool is enthusiastically embraced by HPC application developers, it may naturally find its way into commercial products rapidly.

Selection criteria for determining what activities the center might choose to undertake for specific software tools generated the least debate in the working group. The criteria stressed meeting anticipated users' needs, maximizing the likelihood of delivering the expected results, and moving the state of the tools toward a more interoperable and effective software platform for doing high performance computing. The working group's third key conclusion was that the selection process involve all of the key players (users, researchers, vendors, and center staff) and that individual tool decisions be made in the broader context of a long-range strategic plan for software tool development and use. Once these high-level understandings become clear, the selection criteria appear to fall out in a reasonable and natural fashion.

Intellectual Property Issues

Working Group Members

Intellectual Property Issues	
Donna Bergmark	Cornell
Frederica Darema	DARPA
Jeff Durachta	IBM
Anngie Johnson	NASA

Doug Kimelman	IBM
Tom Kitchens	DOE
Mike Koszykowski	SNL
Bart Miller	Wisconsin
Ken Miura	Fujitsu
Cherri Pancake, Chair	Oregon

Introduction

A working group was convened to discuss the legal/ethical responsibilities, intellectual ownership, and liability issues that would need to be addressed in establishing a national center for software tools associated with high-performance computing (HPC). The group's members included representatives from academic, industrial, and federal organizations involved in HPC. (One participant, Frederica Darema, had participated in the HPSST task force where the concept of such a center was first addressed.) A list of charter questions was furnished as a starting point for the group's discussions.

This summary presents the general scope of deliberations over the three-day period. Rather than following the chronological order topics were addressed, it groups the issues and recommendations in terms of three major topics:

- Why intellectual property (IP) is problematic for the proposed center
- Alternative models for the center and their implications for IP
- Recommended policies for dealing with IP issues

Why Intellectual Property is Problematic for the Proposed center

By its very nature, the concept of a national center implies issues with regard to multiple "owners" of intellectual property. Regardless of the specific products to be delivered by the center, or the specific processes by which they are delivered, at least one of the following conditions will apply:

- the center adds value (e.g., through testing and refinement, additional development, etc.) to software initially developed by individual(s) from another institution,
- the individuals collaborating on a project directed by the center include employees of different organizations or institutions,
- the ideas involved in a project are contributed by individuals employed by multiple organizations, or
- the result of a project is distributed, ported, or enhanced by some organization other than the center.

That is, unless the center designs and develops products from scratch and performs all distribution tasks, multiple organizations will have to be involved. Given the current climate for IP, that in turn implies problematical issues of ownership and liability.

The center, as laid out in the plenary Workshop discussions, itself constitutes a multi-agency effort, as it is likely to be funded by more than one HPCC agency. Currently, each agency imposes its own set of IP requirements. An anecdote was supplied concerning a joint development effort funded by DOE but involving two national laboratories. The project ultimately foundered because the labs were managed by different contractors, who could not arrive at IP agreements - despite pressure from their common funding agency. The Working Group concluded that it would be extremely helpful to the whole concept of the center if the HPCC agencies could come to agreement on the basic IP issues involved in jointly sponsored ventures (see Recommendations).

The group attempted to identify precedents, among recent efforts in the software development community, for how the center's IP problems might be addressed. The National High-Performance Computing Software Exchange (NHSE) sidesteps basic ownership issues by not distributing software, but rather providing pointers to the original software owners; it essentially functions as a clearinghouse. It was the consensus of the group that this model did not make sense for the center, since it would essentially duplicate an existing effort. Instead, the center should provide some type of output.

NASA's COSMIC effort provides Beta code in return for a nominal licensing agreement, allowing others to test and even modify the software. If the software is taken to product status, however, ownership (including any enhancements added by beta licensees) reverts to the original developers and the licenses can be revoked. Similar examples were cited of companies releasing (revocable) alpha source code, to obtain early reactions from users or in cases where there are no immediate plans for further development of the software. As a precedent for the center, this approach was considered too restrictive, since it would assume that all software would be developed strictly in-house, rather than obtained from, or developed in collaboration with, other groups.

We chose to view the overall process in very general form. The center would accept inputs from some other community (e.g., academia), perform some development or enhancement activities, and release some form of outputs to the user community and/or to the software industry for productizing. In terms of the process inputs, basic IP questions arise:

- Who designed and/or developed the inputs?
- Can they prove (sole or joint) ownership?
- Who indemnifies their work, and to what extent?

The outputs are also subject to some elementary issues of IP:

- Who maintains ownership?
- Who will be permitted to use it, and for what purposes?
- Who will be permitted to redistribute it to other potential users?

- Who will be permitted to derive new tools/environments from this work, and under what constraints?
- Who indemnifies the new work, and to what extent?

Specific ideas for dealing with these questions are outlined in later sections.

The group also considered this process from the viewpoint of liability. Here, it was felt that there are a significant number of precedents establishing the desirability of disclaiming all liability with regard to software correctness or appropriateness when applied to specific tasks. The fact that multiple organizations might be involved does not appear to have particular impact, as long as the center maintains appropriate disclaimers.

Another potential problem is the fact that individuals involved either in developing the inputs, or in the center's own activities, might be privy to non-disclosure information from one or more vendors. The group eventually decided that this was not, strictly speaking, a problem for the center; rather, it was up to each individual to abide by the conditions of the non-disclosure agreements into which he or she had entered.

The issue of "copyleft" was also of concern to the group. Under this type of agreement, the adopter of a software component agrees that any time the new work is distributed, complete source code will be made available. Although this does not entirely preclude distribution of software in binary format, it does require that the software developer (in this case, the center) guarantee that source is freely available. Copyleft can pollute the code, in the sense that it may preclude combination of a tool prototype with certain third-party components (i.e., that put restrictions on source code distribution). It may also make it impossible for companies to commercialize the software.

Alternative Models for the center and Their Implications for IP

As the plenary discussion of the proposed center had left some doubts about its precise role in the development and hardening of software tools, the group spent some discussed several possible models of operation. Where possible, individuals described precedents for the models and indicated the IP issues that had been raised in each case. In all, five likely models were identified. Each is described below, together with our conclusions about the ramifications for IP.

"Consumers' Union" Model

As discussed by the group, this model would constitute an independent review board to test and evaluate software tools. Like the Consumers' Union, reviews would be made publicly available in order to help users determine a priori whether or not a software tool was likely to be of help to them. Two precedents were noted. First, NHSE had originally stated that objective reviews would form an important part of their process. In fact, NHSE has served as a Web-based clearinghouse for information, simply providing pointers to Web pages belonging to industry and research developers, and occasionally serving as a distribution site for shareware. As was pointed out by members of the group, NHSE's decision not to take a more active reviewing role was due largely to IP problems.

Cited as a second precedent was the Parallel Tools Consortium's proposal for a related effort, whereby users would contribute evaluations of current tools. Such evaluations would, theoretically, help other users determine the probable usefulness in new situations. Again, a decision was made not to pursue this project, because of liability problems.

The publishing of evaluation reports is fraught with difficulties of fairness and liability. This is particularly troublesome for software, where there are no accepted standards objectively measuring to what extent a product possessed "quality" or "usefulness". Nor can a new product simply be compared with existing ones, if there is no consistent baseline set for comparison (as is the case with software tools). Essentially, then, any evaluation can be criticized for lack of objectivity.

Further, there is no established test suite against which to exercise tools. The precedent of the Ada validation test suite was cited in this regard. It was noted that, in addition to the availability of a general test suite, the Ada situation involved validation by an independent agency established solely for this purpose. However, test results were not published per se; the public notification simply indicated which compilers had been fully validated (but nothing about their relative performance, nor even how many compilers had failed the tests).

The group concluded that while the experiences of a software tools center might eventually lead to the possibility of establishing sufficient test suites and objective testing techniques, such an approach was not really feasible at the present time.

Independent Testing center Model

Discussion of the Ada validation scheme led naturally to a model where software testing was performed, but the results were kept private (other than perhaps a certification that some tool had passed muster). In this case, tool developers - most likely those from industry - could pay a membership fee or testing fee, submit their software, and receive complete reports as to where it passed or failed the tests.

While the model could be self-sustaining financially, it was not clear to the group why a national center was necessary. It was pointed out that in fact, some universities and centers already function in this role, providing testing and evaluation services for a fee. Overall, the group felt that the model offers little advantage to the HPC user community, since the tool developers are not obligated in any way to apply the feedback in refining their product. Moreover, since test results remain confidential, there is no real way to apply pressure for improving tools.

Standards Definition Model

Under this model, the primary role of the center would be to define and promulgate standards for software tools. It was noted that currently, it is quite difficult to organize standards efforts and bring them to fruition, since none of the HPCC funding agencies seems to view logistical and administrative support for standards groups as being within their project domain.

While the working group thought that the availability of clearly formulated standards reflecting the needs of the broad HPC community might go a long way to improving the tools situation, there were two problems with this model. First, there is no clear indication that this role alone might justify the establishing of a national center (since one agency for standards, ANSI, already exists, even if its record with respect to software is somewhat weak). Second, this approach does not really address the circumstance discussed in the plenary sessions: that new software tools are being generated regularly within the tools research community, but they do not find their way into general use or commercial production. The working group concluded that the role of standards definition would be more likely to succeed if it were rolled into some broader role (the examples of NHSE, Scalable I/O Initiative, and Parallel Tools Consortium were brought up).

Funding center Model

The group also discussed the role of a national center in financially sponsoring tool projects that led to robust, distributable, and potentially commercializable products. Currently, funding agencies support the initial research, and in some cases the development of an early distribution, but so-called software capitalization is rare in the area of HPC tools. Such a role might enable the center to expedite the deployment of tools into the user community.

The primary drawback to this model is the relationship of such a center to its own funding sources. Existing agencies have established peer review mechanisms, and it is assumed that a center would need to do the same thing. In that case, what real value-added would be supplied by the center? As was pointed out by some group members, this arrangement would actually force the HPCC agencies to relinquish some of their direct control over funded projects in software tools. It was agreed that there would likely be problems in terms of both political pressure and prioritization (if, for example, agency priorities did not align with those of the center). We recommended against this model.

Software "Hardening" Model

The fifth model discussed by the group was a mechanism for "hardening" the prototype tools that are already being produced by the research community. The center would accept proposals from that community for projects to test research prototypes, make them more robust through code reorganization, integrate them with other software on HPC platforms, perform tasks associated with user documentation, and in general, convert the prototypes into software "products" that would be distributable, usable, and maintainable.

Acceptance of software as input would be through a peer-review process involving other researchers, users (who would assess likely impact of the tool on the user community), and vendors (assessing likely interest for ultimate commercialization). It was assumed that such prototypes would emanate from a variety of research projects at academic institutions, national laboratories, federal centers, and possibly industry sites (e.g., research groups that have developed software which will not be converted into products). center staff would perform the "software distillation" in collaboration with the original developers, to take advantage of their familiarity with the technology and existing code organization. In some cases, external components - such as parsers or other code analysis modules - might be acquired by the center as library components that could be shared among multiple software hardening projects.

The output of the center, then, would be distributable releases of software tools. The group felt that the primary target audience should be the HPC user community; that is, that users should not have to wait for industry to pick up and productize a tool before it can be used. Therefore, the center would also fulfill a role of supporting and maintaining its output. In addition, the center would attempt to feed its products into the software industry, entering into agreements with companies that wish to create proprietary products. Finally, the center's public distributions might also lead back into the hands of tool developers, who could base derivative works (e.g., more specialized tools, problem-solving environment components, etc.) on the earlier releases.

This model exhibits all the problems associated with multi-party ownership of IP, but the group also felt that it did the best job of responding to the objectives of the HPSST task force, as well as guaranteeing impact on the HPC user community. The group then proceeded to discuss how IP could be managed under such an organizational model.

Recommended Policies for Dealing with IP Issues

The group formulated a number of recommendations for how liability and ownership issues should be handled for a multi-agency center. They fall into the categories of: general policies, ownership, liability, and ethics.

General Policies

The HPCC agencies should adopt consistent policies toward basic IP issues, in order to make joint projects practicable.

The current situation, where each agency defines and manages its own IP process, leads to confusion and in some cases can actually preclude collaborations among developers. Regardless of whether or not a national center is established, we strongly recommend that attempts be made to align the agencies' current practices, at least in the realm of multi-party projects. Such policies should address the issues involved when other groups alter or extend previous work, as well as development efforts that originally involved multiple parties.

The procedures and policies established by the center should make it possible to exploit multiple outlets for center products, including the user community, commercial adopters, and tools researchers.

While it might meet the objectives of the original HPSST task force to simply funnel robust tools to HPC users, this imposes a long-term burden of maintenance and support on the center. This should be the primary target, but IP policies should also make it possible for the center to approach industry about tool commercialization. Finally, where appropriate, it should be possible for other tool developers to acquire source code in order to extend, enhance, or integrate the center's product into other tools and environments.

Given the generally poor understanding of IP issues among the tools research community, one role of the center should be the dissemination of practical information to researchers on how they should document ownership of software.

Ownership

1. In submitting a software project proposal to the center, the author(s) must furnish all relevant IP information, including: (a) a clear ownership "audit" indicating all persons or agencies involved in design or development, up to the point where software arrives at the center; and (b) any restrictions on distribution that would be imposed by the submitters, their funding agencies, or other authors whose work is included in the submitted software.

Proof of IP ownership will be necessary before the center can enter third-party agreements with groups to commercialize the software. It may also be important in case of disputes over the origins or ownership of software distributed to the user community or developers of derivative works. Since distribution is the motivation for establishing the center in the first place, it simply does not make sense to accept software whose ownership can be disputed.

Any restrictions whatsoever on software use - including licensing fees, mandatory registration or tracking of users, royalties, copyleft, limitations on source code distribution, profit splits in the event of future commercialization of the software, etc. - may hinder the center's ability to use the results of projects. In general, we recommend that accepted software be free of any restrictions, although there may be occasions where one or more of the software components are subject to some conditions (see ownership profile).

If the submitted software makes use of components from other authors, these may be subject to more stringent restrictions (in the case of copyleft, the software in which such a component is embedded is also subject to restrictions). It may be desirable to eliminate such elements prior to accepting the software as a center project.

2. Before accepting submitted software, the center will negotiate the specific conditions for transfer of rights with the author(s).

Clearly, the most useful negotiated agreement would be for the center to have unlimited redistribution (subject to export regulations, of course). However, since research prototypes often make use of components furnished by other parties, some types of distribution or redistribution may be precluded by agreements already in effect. Any limits that original authors or their agencies will impose on redistribution, licensing, registration, etc., must be specified and agreed upon at this point - prior to accepting the software as a center project (see ownership profile).

It should be made clear to the authors that no ownership agreement guarantees that the software will be released, or if it is released, that it will be supported or maintained by the center for any period of time.

3. Contracted staff should do all development work on center projects, so that ownership of new code or interfaces resides with the center.

Such staff need not be permanent employees of the center, since work-for-hire still retains ownership for the center.

4. If software components are acquired by the center from other sources, they will be subject to similar considerations and procedures for managing ownership.

Certain key components (e.g., a parser) may be acquired from third parties, not for refining but for use in other center projects. Such acquired components add a new thread of ownership and require an ownership audit and negotiated transfer of rights (see ownership profile).

5. The center will require software submissions to be in source form although it may restrict ultimate distribution to binary formats.

The issues are too complex if the center is not supplied with source, or if the original author restricts the center's output to certain formats. Moreover, such arrangements are likely to preclude commercialization agreements. In the case of third-party suppliers of components, however, the center may agree not to distribute source code (see ownership profile).

6. The tools produced by the center will be distributed as licensed products in order to facilitate the acquisition of information on usage and adoptions.

In general, the center should make use of the simplest licensing mechanism, "shrink-wrapped licenses," for non-commercial use of its software (commercialization is discussed separately). Tools that are hardware- or system-specific (e.g., those relying on information supplied by some profilers or compiler code generation) may raise specific problems for distribution.

7. In some cases, the center may allow third parties to redistribute its products; such distribution will be permitted for binary format only, and must involve no fees or special registration

procedures.

From the perspective of the user community, it would be desirable that computing centers (e.g., the NSF centers or national laboratories) be able to pre-build binaries for quick downloading. This may also be to the center's advantage, since it might off-load user support functions. One drawback is that the chain of user registration will be lost, as it is not feasible to require such users to register "backward" to the level of the center (see ownership profile). However, re-distributors will be asked to establish suitable mechanisms for tracking or estimating usage.

In some cases, the negotiated agreement with the original author may preclude such re-distribution, or may impose additional restriction on how it may occur.

8. If a center product is released for commercialization, a separate agreement regarding ownership will be negotiated with the new party.

It is reasonable to expect that the center might be able to recoup some of its costs by effectively selling some of its ownership rights. In most cases, however, such sales should not force discontinuance of the center's rights to distribute its product to the user community. That is, the commercial product should be considered a "new tool," owned entirely by the commercial developers, rather than a derivative work.

In some cases, the negotiated agreement with the original author may preclude commercialization, or may impose additional restrictions on how it may occur. If a profit-split was specified in that agreement, the original authors may need to be included in the price negotiations.

9. Derivative works based on center products will be permitted (subject to appropriate credits) and may be distributed to any registered recipient of the original source work.

This is intended to apply primarily to center output that can be distributed in source format (see ownership profile). If derivative use of binaries is to be permitted, special modification of the shrink-wrap license may be required to specify how attribution of credit is to occur.

In some cases, the negotiated agreement with the original author may preclude derivative work, or may impose additional restrictions on how such derivatives may be distributed.

Liability

1. In keeping with the current policies of most software vendors, the center will not accept liability for defects in judgment, engineering, use, etc.

The now-familiar disclaimer of any responsibility whatsoever should be attached to all center output.

Ethics

1. Acceptance of a prototype obligates the center to certain responsibilities toward the original software developer(s).

In furnishing software to the center, the author is ceding direct control over its future, with the expectation that it will be prepared for release to a broader user community. Consequently, the center should periodically inform the author of the software's status. If a decision is made to drop a project, the center is obligated to inform the author as to why, and all rights (including any value-added) should revert to the author. If a decision is made to discontinue support of a completed, distributed project, the center is obligated to inform not just the author (with similar reversion of rights) but also the community of registered users, so that they have the opportunity to archive the last available version prior to its disappearance from the center's inventory.

Conclusions

The group considered five alternatives for how a national center for software tools might operate, in terms of the IP issues and problems associated with each. Only one model appears appropriate, given the objectives of the original HPSST recommendations and the current national climate with respect to IP ownership and liability.

The consensus was that it would be inadvisable to establish a center directed at testing the quality of software tools and making the results available freely to the general public. The lack of widely accepted testing and evaluation standards, and the associated liability issues, make it unlikely that any institution or organization would be willing to stand as publisher of the results.

Restricting distribution of the test results to just the tool developers would eliminate most of those problems. However, the group deprecated this model on the grounds that the potential benefits for users would be seriously diminished, and that a private facility or consortium would be a more appropriate vehicle. A model focusing on actual tool implementation was rejected as superfluous, as this approach would replicate ongoing, peer-refereed programs by the HPCC agencies. A fourth model, where the center's output would be definitions of standards for software tools, was rejected as not meeting either the intent of the original HPSST task force or the needs of the tools research community.

On the other hand, the group concluded that it would be possible to formulate IP policies that would allow the center to follow a fifth model, that of software "distillation" or "hardening." The center would acquire software in varying stages of development, refine and test it to ensure its robustness and suitability for HPC, and release it (not just to the user community, but also to potential adapters/extenders as well as commercializing groups). Examining this model, the group formulated a series of recommendations on how such a process should be structured to deal with IP issues. Due to growing concerns about software and its relationship to copyright and patent law, it is essential that the center insist on strict documentary evidence of ownership before accepting software from any development group, particularly when development involved multiple agencies. Most other aspects of IP can be negotiated on a case-by-case basis, although clearly the ideal situation would be software that the center can distribute or re-use freely, without the need for cascaded licensing or other restrictions.

Nevertheless, the model presents some potential problems. For example, depending on rights agreements, the center may accumulate an inventory of components, each with a pedigree and new licensing agreements/restrictions. The licensing and ownership rights could end up being extremely complex. Is this self-defeating? Most importantly, would this work against the primary goal of getting tools into the hands of the user community (direct to users)? Another example is the problem of what responsibility the center might have toward the groups that contribute software. If long-term support and maintenance is expected, serious problems could arise when software becomes outdated or is superseded.

In summary, the group felt that with a proper structure - and careful explanations to the tool development community - a center could be an effective mechanism for expediting the deployment of new tool technologies into the user and vendor communities. Special care, however, will have to be taken to ensure that IP complexities do not undermine the center's effectiveness.

Charter Questions

- What are the issues associated with the intellectual property right of software tools that a Testing & Evaluation center will have to address?
- What are some possible policies that such a center might put forward to deal with these rights? How might they (the policies) be enforce so that fairness for both the center and the developers is maintained?
- Who owns what if a piece of software is first prototyped by an academic, extended by the center, and integrated with vendor software?
- When are the decisions made and by what process?
- At what point does something become a "new tool"? How long does original ownership last?
- To what extent is the T&E center responsible for claims concerning a tool? Liable for errors in judgment or shortcomings in the T&E procedures?

Technical Infrastructure for a National Software Tools center

---

Working Group Members

Technical Infrastructure for a National Software Tools center	
Don Austin	NCO
Rod Oldehoeft	DOE
Dan Reed	Illinois

Thomas Sterling (Chair)	Caltech
John Toole	NCO
Bob Voigt	NSF

Introduction This working group addressed the technical infrastructure issues associated with the formation of a national Software Tools center. Several working assumptions underlie the observations and recommendations of this report:

- Advances in parallel architectures have not been matched by the needed improvements in software capabilities. As a result, applications development usually proceeds via "heroic" efforts instead of normal development practices generally followed for software on sequential, vector, or smaller-scale parallel machines.
- In spite of the small niche occupied by truly high-performance computers, the applications uniquely possible on these systems are extremely important for advancing science, developing industrial competitiveness, and supporting defense needs. Hence the software difficulties in this niche must be addressed as a national priority.
- New approaches are being developed among the research communities in universities and national laboratories for high-performance system and tool software. Often these proofs-of-concept show exciting potential for improving the current situation. However, these communities have primary responsibilities to their applications, or to software research as opposed to production. The resulting software is often characterized as rudimentary, brittle, poorly documented, and isolated from other software. As a result, these are not widely adopted and, worse, are often re-invented for other sites and new projects.
- A National Software Tools center will be useful in providing the means for selected experimental codes to be transformed to usable robust software tools. The initial codes will be derived from research groups around the country. Multiple stages of maturity will be specified and target codes identified for transition across successive stages. As understanding of the potential value of each maturing tool grew through use of early releases, additional resources will be applied to continue the advance, possibly to the point of full commercialization. The nature and structure of such a center has yet to be established.

This report documents the findings and recommendations from working group deliberations. The following sections provide discussions about center objectives, the products from the center, criteria for how projects may be selected, characteristics of the center, resources required, and the recommended skills mix for center staff.

Objectives and center The purpose of the proposed National Software Tools center (referred from here on as "the center") is to dramatically enhance the state and utility of high-performance computing through increased availability of essential and advanced software tools. The term "tool" is applied here in the broadest sense and refers to any software that extends the capability, usability, and

Inputs

efficiency of high-performance computing systems in the development and performance of end-user applications. Such tools include but are not limited to compilers, run time systems, operating system components, debuggers, performance profilers, and tools for data management, scientific visualization, communication, fault management, and software integration. The global vision behind the center is to provide the necessary infrastructure to select appropriate software outputs from research projects and carry them forward to a form suitable for use by the general HPC community. This includes the possibility of creating software products not being addressed by any research groups. The center is not necessarily to be a single location, but may engage talents at diverse geographical locations and across administrative domains.

Activities at the proposed center will be driven by several inputs:

- A major driver will be the early proof-of-concept codes from experimental projects in software tools research. These may come from any cooperating research organization including universities, national laboratories, other not-for-profit research institutes, and even computer vendors. However, no source may impose proprietary considerations that would limit availability of any center results to the general HPC community.
- A second somewhat different source are prototypes of tools assembled by users where the primary goal is the application, but where ad hoc efforts produced inchoate tools. This is an example of a broader input source: requests, explicit or implicit, from the user community for tools that satisfy recognized needs. This class of input may be source code or precise functional specifications.
- Finally, the center staff will also identify needs in-house that require development of new tools. Here, only a specification instead of an initial test code is available at the outset of the resulting project.

Products of the center

The end products of the center will undoubtedly take on many forms, resulting from the varieties of inputs, and the functionality desired. Therefore, a scale of intermediate products will be supported by the center. While there is no intent to constrain the product types or the degrees of robustness the center's products may have, there are some major identifiable categories that typify the kinds of tools likely to be produced. These can be distinguished by completeness or assumed reliability.

1. Early Evaluations

An initial but important product of the center is the early evaluation. The result is not usable code, but instead a detailed critique of the merits of the concept, approach, implications, and implementation methods exhibited in an early design or prototype. This is a formal process that makes available to another group an objective and detailed assessment of a new project, and its potential for addressing key problems within the HPC software tools arena. These evaluations will be made available to the research groups involved to assist in guiding and influencing direction at an early stage. Among other contributions, the center can alert researchers to other work in the specific field they are pursuing and compare the intended contribution with other efforts. This particular class of product from the center enables the center to assess the potential merits of some future collaboration with the target research group and project.

2. Improved Prototypes

Perhaps the single most important product of the center will be the improved prototypes of research codes and tools. Indeed, this was the original idea that sparked the genesis of the center and would alone justify its creation. The intent of the improved prototypes is to bring the potential functionality of research experimental codes to a high enough level of reliability that they can be used by a select "friendly" community to use and evaluate. The improved prototypes, while not bullet-proof, will be expected to work as intended under most operational circumstances. The center will modify and augment the original research tool until it reaches center standards of quality for prototype release. Documentation for installation, interface, and use will be an important element of the improved prototype. In addition, test cases will be provided to end users to determine correct operation after installation. The center will provide support, tracking and fixing bugs as reported in the field. The center will also establish, for each improved prototype, a user evaluation and reporting database to collect assessments from users. This information will determine future efforts towards further improvement, as well as continued support. Many levels of "improvement" will be possible by the center, thus providing flexibility in taking on new projects and meeting the needs of the community. This will permit better use of center resources, allow more experimental tools to be engaged, and reduce the necessary level of effort to moving a given prototype tool to the next stage of development. It will be the responsibility of the center to establish a framework for defining these levels of improvement and procedures for managing tool development through these successive stages.

3. Reference Implementations

A few research software tools will prove to be of high enough quality and value that their widest possible distribution will be imperative. Possible commercial implementation by major system vendors or independent software vendors might be appropriate and the ultimate goal. To support the commercialization as well as the early availability of such codes, a high level of robustness, specification, and documentation will be achieved in a center product to provide a reference implementation. A reference implementation is a self-defining specification of functionality and interface as well as a fully operational tool. Users of reference implementations of tools from the center can expect them to be of high enough quality to be used on a production basis and can install them among their main software tools. Once a tool has reached the level of reference implementation, additional changes to functionality will be rare and will be reflected by controlled version numbers; this to retain uniformity of different vendor implementations and manage user expectations.

4. Conventions and Standards

A major challenge to the software tools community has been the collection of commonality characteristics that enable portability, interoperability, generality, and functional uniformity including user interface. The unfortunate alternative is a collection of isolated and unrelated tools unable to exploit the capabilities of others or, in ensemble, function as a higher-level complex system. To support better code reuse and to enable tools that exploit other tools' capabilities, a set of interface standards will be devised by the center to specify conventions for interplay of tools. Tools crafted to comply with such standards will be more easily integrated into a powerful and evolving ensemble. New tools will be fabricated more quickly because developers may reuse existing and accessible functionality. Developers will realize a larger immediate user base as the community more readily adopts compliant tools. Such a set of conventions will expose gaps and opportunities for future advancement of capabilities in much the same way that the Periodic Table, once incomplete, exposed plausible but undiscovered chemical elements. The center, out of necessity, will develop conventions for interoperability of the tools built in-house from research codes. De facto standards will sometimes be established as guidelines to future development. These will be shared with the HPC community and, where appropriate, used by the community in general.

5. Education

Even without specifying its characteristics, the center will clearly be well positioned to play a role in education. It will surely provide instructional mechanisms and materials related to the software with which it is involved. Beyond that, it may contribute material related to the education of future computational scientists and users of high-performance computing systems and tools. Possible forms are varied: tutorials for the use of HPC tools; curricular elements developed with educational institutions for preparing future computational scientists; books and other documents focused on this narrow but important field. Defining the exact role for the center to play in education is a task for future inquiry.

Project Selection

An important aspect of center operations is the actual selection of specific projects to be undertaken, driven by the many opportunities provided by the research communities, and the needs of the HPC user community. If successful, the center will have positive influence over the evolution of HPC software tools, based on which projects it selects to advance by applying its resources. This significant responsibility demands selection processes and criteria that both represent and support the research and user communities. Criteria will unavoidably conflict, and difficult choices for expending limited center resources are inevitable.

Among many possibilities, several criteria are identified for discussion here.

1. Strategic fit with center objectives

The objective of the center is to put the best tools of greatest importance into the hands of users as rapidly as possible. Underlying any possible selection process is a basic center strategy or model of what composes an effective software tool set. This is in turn driven by a conception of the requirements and state of the art. Proposed projects which most closely fit this model are more likely to be sponsored and actively pursued.

2. Potential impact

Within this conceptual framework, the potential short term and long term impact will be assessed. A major driver of the selection process will be those factors that are expected to deliver the greatest ultimate value to the research and user communities.

3. Innovation

Innovative concepts and approaches are critical for rapid advance in this emerging field. The more radical or advanced the approach, the more likely it is to contribute to establishing new paradigms for managing HPC system resources. However, innovation must be tempered with practical considerations of utility and compatibility. Nevertheless, the benefits of novel tools may outweigh the inconvenience and disruption to conventional but less productive user practices. The most valuable tools will be those which provide new functionality that fill recognized needs, but which complement and interoperate with the community's existing base of tools.

4. High quality of incoming software

The level of effort required will be a strong function of the quality of the original research code to be enhanced. Where good software engineering practices were used in the development of the initial experimental tool software, the likelihood of success, with lower center investment, is enhanced. Conversely, a large morass of undocumented spaghetti code would require much more center effort, and so is less likely to be selected. Level-of-effort concerns also favor small, modular projects instead of grandiose "we've solved the whole problem" projects. In part, the conciseness of the project objective is likely to be reflected in well-crafted codes instead of huge software dreadnoughts.

5. Maturity of prototypes

Of course, level of effort will also be sensitive to the degree of the work already achieved by the originating research group. The more advanced the effort, the easier it will be for the center to transform it into a robust prototype. Further, the confidence in capabilities and potential impact for an advanced project is enhanced, which can make selection more likely. More mature codes will have had more extensive use and evaluation, the results of which will influence the review and selection process. Codes capable of strong and rich demonstrations will be favored over early breadboard codes that have been exercised in limited ways.

Naturally, there must be a balance here. It can not be the unintentioned implication of the selection process that research groups are forced to do the job that the center was established for in the first place. But where choices must be made, those projects most likely to lead to success for the HPC user community can be expected to be favored. That includes the confidence and level of effort in the project, both of which will be influenced by degree of completeness of the original research tools code.

6. Potential for fruitful interaction with producer groups

It will not be practical for center staff to be fully versed in every detail of the initial research code. Its quality must in part be surmised from knowledge of the originating research group and its past accomplishments and products. This may appear to favor the well-established and better-funded research groups, which is certainly not the desired outcome. It is, however, reasonable to favor projects from groups with strong and productive track records. As graduate students and postdoctoral researchers from successful groups diffuse throughout the broader community, their reputation will follow them to raise the standards across the domain. A long term consequence of this necessary bias is that the standards of code quality will rise not a bad thing in the long run.

These criteria have differing weights when applied to the several classes of center products.

The objective of Early Evaluation projects is to accelerate the advance of innovative ideas in constructive directions. The focus will be on inchoate projects which may have less experimental code to demonstrate but novel yet solid concepts to present. The low level of effort required by the center to assess the merits and provide constructive recommendations means that the selection criteria will be more heavily weighted toward potential importance as well as quality of documentation than toward other criteria. It also provides an early look at a particular project at a time when direction may be strongly influenced. Any project that has gone through this process is more likely to be selected at a future time as a target to improve the early prototype by the center, as it will more likely reflect the center's basic model.

Projects selected for developing Improved Prototypes will be subject to additional considerations, including the continued participation of the producers. A close relationship between the developers and the center is essential for successful technology transfer to the center. The complexities of incompletely documented, highly experimental code will make code understanding dependent on tight collaboration with the producing group. If such a relationship is not feasible, the project will be less attractive as a target for prototype improvement.

Another factor in the decision will be the kind and degree of improvement necessary to bring the code to the next stage of utility. This, combined with its potential impact related to functionality, will determine how quickly a new useful tool can be brought to the community.

Reference Implementation projects will be selected according to the criteria presented above, but include additional factors associated with the likelihood of vendor participation. The value of a reference implementation is in a de facto standard of functionality, so that vendor implementations will achieve identical interface, interoperability, and equivalent behavior. Robustness, completeness, internal consistency and specification documentation are critical characteristics of a center project intended to serve this role. The significance of the responsibility implied by these requirements dictates that a high level of effort will be required by the center. Few such projects are anticipated, and certainly no more than two of them is expected to be engaged concurrently by the center. Therefore, selection will be stringent and depend on a high probability of success. A key component to that will be the participation of the vendor community in its evaluation and endorsement of the end product. One or more vendors will be required, a priori, to show strong willingness to consider internal advanced development and product distribution if the project is to be undertaken by the center. This will mean that the contribution to be made by releasing the reference implementation is clear and compelling. Such evidence will come from use by parts of the community of earlier advanced prototypes of the research code previously developed and released by the center. Additional issues of ownership and reference version control must also be resolved before project initiation.

Standards and Conventions are of a different nature than the other center product types. These are frameworks or conceptual infrastructure that enable software tools communities and their products to work together and to provide a necessary level of stability to the end user community. Selection of efforts to establish such standards or conventions will be derived from perceived need both within the center and by the community. They will emerge, sometimes unintentionally, as a natural consequence of just trying to get the center's jobs done.

Selection will involve a mix of contributors. The primary and final decisions will be made by center management. An external advisory board will be established from representatives of the sponsoring agencies and the HPC community. This continuing body will advise on all selections made, especially in establishing priorities and tradeoff criteria. Vendor representatives will be consulted for selection of projects to develop reference implementations or standards. The selection decision process will involve two stages: merit for selection, and priority for resources. The first stage judges the candidate project on its own intellectual and functional merits. The second stage determines its competitiveness relative to other potential projects and finite center resources.

---

center Characte-

A close relationship with the HPC tool developing and using communities is crucial to the center's success in advancing the state of software tools and high-performance computing. Access to external talents in both domains is critical to extending the effective capabilities of the center beyond those encompassed by the immediate center R&D staff. Hence the center

ristics should be co-located at a national HPC host site such as a DOE national lab, an NSF supercomputer center, or a NASA center with strong computational programs. Another reason for establishing the center in such a context is the availability of several large computing systems, and the presence of an independently maintained infrastructure. Thus immediate and easy access to end users, expertise, and resources strongly supports hosting the center at an established HPC institution.

While the center will be co-located at an HPC host site, it must be independent of the hosting institution with regards to management authority and mission direction. The center will not be perceived to be owned or unduly influenced by the host. Rather, the relationship can exploit the potential synergism through mutual exchange of ideas, talents, and resources. The center will reimburse the host site for use of its facilities. Participation in center activities by host site personnel will be arranged on a case-by-case basis and will most likely be unfunded collaborations.

While a single centrally located monolithic organization is one possible form of the center, several considerations lead to an alternate form being adopted. Access to a diversity of HPC platforms is less likely to be achieved at a single site than at a collection of separate sites. For example, the NSF Supercomputer centers collectively provide access to several types of machines among them. Computational centers often tend to focus on specific classes of application most closely associated with the mission of its sponsoring agency. For example, the DOE national labs and the NASA centers involved in high-performance computing are engaged in distinct applications, although many of the underlying algorithmic principles overlap. Finally, the best talents in system software are found among several organizations, not in one place. For these reasons, the center will take on less of a form of a "center of Excellence" in favor of a structure more like a "Circle of Excellence" by distributing the center organization across several geographical sites. It is proposed that the center comprise three or four distinct but strongly coordinated sub-centers, each located at a separate location and host site. This will give access to a diversity of resources, talents, and user requirements, as well as help better focus on the distinct missions of the multiple sponsoring federal agencies.

Of the three or four sub-centers, one will take on the additional responsibilities of administrative and coordinating duties as well as operating as interface to the external sponsoring agencies. However, all sub-centers will engage in the technical process of selection and execution of center projects. Each sub-center will be managed by a center deputy director with a center director having general responsibility for the ensemble. Projects will usually be allocated to a specific sub-center instead of attempting to distribute the workload across sub-centers at a finer granularity. Proximity of co-workers leads to rapid progress and serendipitous discovery. Matching of project to sub-center will be determined by several factors including workload, relevant talents and resources, and possible proximity to the producing research group. Other issues may come into play as well on a case-by-case basis.

---

center Resources The focus of the work of the center is the development, enhancement, and testing of innovative software tools for high-performance computing, so this perforce involves use of high-performance computers. Because the tools under development will often be dangerously shaky or interface with low level mechanisms buried deep within operating system kernels or device drivers, direct and full access to, as well as control of, HPC systems will be required. At the same time, access to large systems will be essential to verify correct operation at scale and to determine scaling properties. While generous funding by sponsoring agencies is anticipated for center functions, it would be impossible and inappropriate for funds to be expended on one of every kind of HPC machine in its largest possible configuration.

This conflict of needs and realities will be satisfied by a mix of small, program development systems being acquired and placed at the sub-center sites for exclusive use by center technical personnel. These development systems will not be expected to provide a robust and uninterrupted user environment, but can be the target of disrupting low level system software and tools development efforts. It is expected that each of the major vendor platforms will be represented by one of the sub-center development systems. No two sub-centers is expected to have the same class of development system and therefore a means of distributed sharing across the center is essential. Such mechanisms and system administration issues to make this both possible and easy to use must be established by center management.

The need for access to large configurations of HPC platforms will be satisfied by the host sites. Each such site will be selected in part for its large high-performance computing facilities. The locations of the sub-centers will chosen to maximize the diversity and size of the host systems available. Although the small development systems will incur most of the down time resulting from the experimental development and testing cycle, there will be periodic requirements for the entire host site system to be made available to center project teams. Support by the host site of such intrusive activities will be an important criterion in site selection for sub-centers.

In addition to the small development HPC platforms, the center and its constituent sub-centers will own several other computing resources to enable their missions. A heterogeneous collection of workstations will be procured and updated over time. These workstations are necessary both to support the day-to-day computing requirements of the personnel and to provide test platforms for code under development. Many software tools engage both HPC systems and user workstations, sometimes in complex ways. Occasionally, software tools may interact with proprietary products of specific workstation vendors. Graphical user interfaces (GUI) to HPC software tools are usually executed on user workstations. Also, an important class of high-performance computing systems is "clustered computing" using ensembles of loosely coupled workstation-class systems. Thus an important and integral element of the sub-center environment will be its rich collection of workstations.

Other important support resources include a high-bandwidth network, file servers, backing store (tertiary storage), printers, and Internet connection. To some degree, the sub-centers may be a customer of the host site resources to partly satisfy these requirements. In other cases, the sub-center is likely to own the resources it uses. These decisions will be made on a case-by-case basis. For example, it is anticipated that the sub-centers will have large data storage requirements for handling data sets resulting from software experiments and other aspects of center operation. This requirements may be best satisfied by center ownership.

Besides substantial and diverse hardware resources, center objectives require a substantial base of software resources as well. Commercially available software development tools will be an essential component of the software base both for direct use by developers and as targets for interoperability of experimental tools under development. Source code for target machine operating systems and compilers will be critical for providing direct access to low level functionality in the support environment. Mechanisms for accessing protected resources, such as hardware counters, are essential for development of certain types of tools.

Finally, each sub-center must be independent in meeting its daily operational requirements. This implies the need for full environmental support for managing paper work, organizing meetings, presenting material, and providing the usual personnel support functions.

Administrative and secretarial resources must be provided in sufficient quality and quantity that technical staff are not distracted from their principal occupations.

---

center Staff Skills Mix

Management of the center will be limited to essential functions for directing the processes of the center, administrative and logistical support, and interfacing with the HPC community and host sites. Overall center management will be provided by the center Director, who will be primarily responsible for coordinating with the center Advisory/Steering Committee, establishing direction and procedures, and maintaining relations with sponsoring agencies. The center Director will also make all final decisions about new project starts and continuation of on-going projects. However, these decisions will be made in consultation with the review process of the center Advisory/Steering committee and based on recommendations of the center technical staff. Every sub-center will be under the direction of its Deputy Director, who will be responsible for the operation of the sub-center, relationship with the host site, and the progress and quality of the projects. The Deputy Director will be supported by the Chief Administrator and the Chief Scientist of the sub-center. The Chief Administrator will manage the administrative support staff and all budgets. The Chief Scientist will oversee all technical projects of the sub-center as well as conduct specific projects.

The principle function of the sub-center will be the development and enhancements of HPC software tools; this will be carried out by the sub-center technical staff. At least 50% of all personnel will comprise the technical staff. A mix of expertise and capabilities will be represented by the permanent technical staff. Such backgrounds will include operating systems, compilers, GUIs, scientific visualization, evaluation and instrumentation, computational science, parallel application programming, and modern software development practices. Projects will be conducted by teams of members of the technical staff. Each project will be directed by a team leader who will be dedicated to that task. However, while most members of the team will be focused on the single task as well, individuals with specific talents critical to more than one project may be shared, workload permitting. It is paramount that all members of technical staff be well versed in modern software practices. Training in this area may be required and provided for new hires.

Staff will be necessary to provide important support services. This goes well beyond the typical secretarial support ordinarily found in any organization. Because of the importance of the complex computing facilities accessible from the sub-center, substantial systems administration personnel will be located at every sub-center. These people will be challenged by the conflicting needs of providing robust capabilities while making systems available for risky experiments likely to cause individual systems to fail while under test. The center will be responsible to the user community for the software tools it releases. A permanent and well-staffed user help desk will be supported by each sub-center to maintain the software tools and provide advice to users. The user help desk will interface to the user community for managing all bug reports and providing rapid response when possible. These staff will constantly be learning new tools as they are developed and will work with the technical staff as software tools are being readied for release as advanced prototypes. Documentation is essential for conveying functionality, interface requirements, and means of use of new software tools. The sub-centers will include permanent technical writers on staff who will work closely with code developers to provide the necessary documentation to the user community. The center, although not every sub-center, will engage the services of legal counsel on a continuing basis to deal with issues of ownership and liability related to experimental software tools.

A significant number of people at a sub-center at any particular time will be visitors, for several reasons related to the objectives of the center. Users from other institutions will visit to convey needs and to assess the merits of emerging software tools. Original developers from other groups whose codes have been selected for center projects will be on-site to help in technology transfer both for explaining the details of their code and in receiving critique related to the merits of their codes. Consultants with specific expertise necessary for a given project will be housed at the sub-center. Representatives from industry and vendors will be on-site to work with tools developers, especially in the case of reference implementations. Students, postdoctoral researchers, and faculty on sabbatical will be important visitors to enrich and diversify the interests and capabilities of the center community.

Technical Issues – Debugging

- Working Group Members
- General Discussion
- Recommendations
- High Performance Debugging Standards Forum

Working Group Members

Technical Issues -- Debugging	
Gail Alverson	Tera
Jeffrey Brown	LANL
Karla Callahan	Intel
Suresh Damodaran-Kamal	Hewlett-Packard
Erica Dorenkamp	Thinking Machines
Joan Francioni (Chair)	SW Louisiana
Mike Gittings	LANL
Kenneth Koch	LANL
David Metcalfe	SGI/Cray
Juan Meza	SNL
Lauren Smith	NSA
Rich Title	Hewlett-Packard
Abduhl Waheed	Michigan State

Introduction

This working group was given the task of discussing issues related to debugging, testing and verifying the correctness of paral lel programs used in the high performance computing arena, and making recommendations along these lines with respect to the role of a national tools center. In particular, the group was asked "What should be the role of a tools center to encourage the development and accelerate the deployment of high quality parallel debuggers for high performance computing?" This document summarizes the group's responses to that question.

General Discussion

The goals of an effective debugging tool for high performance computing are for the tool to be useful in helping users find out what they need to know to understand and debug their programs, and also for it to be portable, extensible, scalable, and easy to use. The debugging community has yet to come up with one debugger that meets all of these goals. This is not because the goals are not understood. Rather, it is because debugging tools are very complex pieces of code that must be written to run on a specific machine with a specific compiler and under a specific operating system. As these three components are constantly changing in the high performance computing arena, it has been difficult for researchers and vendors to have access to stable platforms long enough to be able to develop appropriate tools. Also, there are no accepted standards about what debuggers should do nor about what compilers/operating systems should provide to the debugging program. This greatly increases the complexity of writing a debugger from scratch for each new system.

The problems users are experiencing with current tools range from nonstandard semantics of the commands (e.g., breakpoint and next) - to confusing screens full of too many windows - to an inability to do the things they need done. The effort required to learn a new tool has also frequently turned out to be a problem for users. It is not cost effective for a user to spend a large amount of time learning a new tool for a platform that may not be in existence for very long or for a tool that is not very useful. In addition to these types of problems, both users and developers cite the lack of user experimentation and input for tools under development as a serious impediment to building useful debugging tools.

Recommendations

The working group considered the overall purpose of a tools center to be two-fold: (1) to support the development of useful technologies and research ideas; and (2) to support the promotion of effective tools back to the user community. To this end, five goals for the center were defined. These goals are listed below along with possible functions a tools center could facilitate. In some cases, functions appear listed under more than one goal.

1. Increase understanding of the needs of both users and developers from each other.

There is a definite lack of communication between users and tool developers that has been hard to overcome. The need for this communication is well accepted, but many strategies used in the past have been less than successful for a large number of tool development efforts. The PTOOLS consortium is definitely attacking this problem, but more can be done. Possible functions for a center related to this goal include education programs/workshops, usability testing, developing and promoting standards, and establishing mechanisms for user feedback of tools throughout their development.

2. Support proof of concept for promising but immature research software and ideas.

A good debugging research idea can easily be prototyped, written up and published, but that doesn't tell us enough about how the tool/idea will work "in the field." Conversely, most academic researchers do not have the resources to develop full-blown, robust software to test out their theories on a large slate of real applications and/or with a group of real users. Thus, achieving this goal requires identifying "promising" software via early evaluation of research ideas and then providing support to further develop the software so that it is robust enough to go through usability testing. Related functions would include porting the code to other platforms and developing a base of code building blocks for constructing prototypes quickly.

3. Facilitate testing and evaluation of tools.

For both vendors and academic researchers, having access to the large/realistic systems that the users actually use has been a problem, particularly in the testing and evaluation phases of tool development. In addition, there is no accepted suite of programs that can be used for testing the functionality and performance of debugging tools. Functions which could be provided by a center related to this goal include usability testing, developing and promoting standards, and scalability testing. It would also be appropriate for the center to develop a test suite and a set of benchmark programs for evaluating new tools.

4. Reduce effort needed for academic and vendor developers to create usable tools.

This goal is about establishing an effective mechanism for getting user feedback and input to developers at appropriate stages of a tool's development. It includes the functions of developing and promoting standards, usability testing, scalability testing, developing benchmark programs, and establishing a code base.

5. Support useful tool base for user community.

The intent is not for a center to compete with system vendors or independent software vendors. Rather the goal is to help establish a critical mass of users for effective tools. Once a useful tool has enough dedicated users, it can then be supported by the vendors. (We consider a recent example of this model to be the development of PVM.) In particular, a tool would only be supported by the center for a finite amount of time. If it fails to be picked up by vendors by that time, it would be dropped. Possible functions in this category include usability testing, benchmark programs making tools more robust, and porting code to multiple platforms.

A common thread during our discussions, and reflected above in the function lists of each of the goals, is the need for usability testing by a representative group of real users. This is considered a critical function for making any headway in developing useful and cost effective debugging tools. In addition, user testing should be done early and repeated throughout the entire development of a tool. Although this is a critical function, it is not an easy thing to accomplish, especially in an ad hoc manner. Users are busy working on their own problems and it is not considered part of their job to spend a large amount of time testing out new debugging tools. Tool developers frequently do not have enough connections with a wide audience of users to get appropriate input and feedback throughout the development of a tool. A nationally funded center, however, should be able to set up and support a mechanism that will allow real users to try out different tools and provide expert guidance back to the developers so that the developers can spend their efforts on designing effective and useful tools.

The tools center that is developed may address some subset of the five goals listed above. Within our working group, the users were interested in all goals being met, the vendors primarily supported 1 through 4, and the academics were most interested in 1 through 3.

After defining the above set of goals for a center, the working group discussed how these goals might be met outside of a tools center. A number of alternatives were suggested that included the following. Workshops, such as this one, certainly support goal 1 and could be increased. A High Performance Debugging Standards Forum could be established (see below) to support goals 1, 3, and 4. More support could be given to facilitate academic researchers spending summer or sabbatical terms at vendor or user sites. This would support goals 1 and 3. The mechanisms in place by funding agencies partially support goal 2, but research funding stops short of the work needed to make tools ready for usability testing. This could be changed somewhat, but it is not necessarily of interest or even appropriate for researchers to do all the detail work necessary to bring a tool far enough along for wide usability testing (i.e., different programming models, architectures, applications, etc.). Finally, it was suggested that a vendor consortium could be established to share ideas and common components, thereby supporting goal 4.

High Performance Debugging Standards Forum

As a direct result of the discussions of this working group, a Birds-of-a-Feather session was held at Supercomputing '96 to explore the community interest in beginning a formal debugging standards effort. It was decided at that session to go ahead with this effort and plans are underway to have a first formal meeting in March in conjunction with the SIAM conference. A steering committee for the forum has been created and consists of Jeffery Brown, Joan Francioni, and Cherri Pancake. In addition, funding for the initial efforts of the forum is close to being secured.